E.T.S.I. DE TELECOMUNICACIÓN
DPTO. DE TEORÍA DE LA SEÑAL Y COMUNICACIONES
Universidad Carlos III de Madrid

# TESIS DOCTORAL

# MÉTODOS DE ASIGNACIÓN DINÁMICA DE INTERVALOS DE TIEMPO PARA REDES DE COMUNICACIONES TÁCTICAS MILITARES

**Autor: Antonio Jesús Vera López**

**Director: Dr. Antonio Artés Rodríguez**

Leganés, Julio de 2006

Tesis Doctoral:

# MÉTODOS DE ASIGNACIÓN DINÁMICA DE INTERVALOS DE TIEMPO PARA REDES DE COMUNICACIONES TÁCTICAS MILITARES

Autor: ANTONIO JESÚS VERA LÓPEZ
Director: DR. ANTONIO ARTÉS RODRÍGUEZ

El tribunal nombrado para juzgar la tesis doctoral arriba citada, compuesto por los señores

Presidente:

Vocales:

Secretario:

Acuerda otorgarle la calificación de

Leganés, a de de 2006

# AGRADECIMIENTOS

En primer lugar, quiero agradecer a D. Antonio Artés el tiempo que ha dedicado como director y revisor de este trabajo, dándome una orientación y guía para esta Tesis mediante las ideas aportadas que han sido fundamentales para su realización.

Debo agradecer también al Laboratorio de Data Links del Departamento de Electrónica del Centro de Investigación y Desarrollo de la Armada (CIDA), especialmente al CN D. Manuel Golmayo y al CF D. Manuel Martínez, las facilidades ofrecidas para poder realizar los cursos de Doctorado y conseguir toda la información y conocimientos sobre las redes de comunicaciones de datos tácticas militares, y más concretamente sobre el estándar Link-16, necesarios para el desarrollo de esta Tesis. También quiero agradecer a Ana Bravo su preocupación y gran ayuda en el desarrollo de este documento.

También quisiera dar las gracias a los profesores de los cursos de Doctorado y a todos los integrantes de los departamentos por los que he pasado, desde el Grupo de Señales, Sistemas y Radiocomunicaciones (SSR) de la ETSIT de la Universidad Politécnica hasta el Departamento de Teoría de la Señal y Comunicaciones de la ETSIT de la Universidad Carlos III, por todo el apoyo prestado.

Finalmente, mi más profundo agradecimiento a toda mi familia, y muy particularmente a mis padres, Antonio y Remedios, y hermanos, Manuel y Belén, por el continuo apoyo y ayuda prestados en todo este tiempo y, sobre todo, por su comprensión en los peores momentos de preocupaciones. Y a todos los amigos que siempre estuvieron pendientes de mis avatares. Gracias a todos ellos.

# RESUMEN

Las redes de comunicaciones tácticas militares tienen como objetivo el intercambio de información entre un gran número de unidades tácticas dispersas en un área, cada una de las cuales tiene un terminal para la transmisión de datos, optimizando las funciones militares operativas conjuntas. Link-16 es actualmente el estándar más avanzado de este tipo de redes de la OTAN. En esta red, a cada participante se le asigna un grupo de intervalos de tiempo (*time slots*) para transmitir sus mensajes. Existen tres modos de acceso a esos *time slots*: dedicado, por contienda y por reasignación de *time slots* (*Time Slot Reallocation*).

En el modo *Time Slot Reallocation* (TSR) se asigna un mismo conjunto de *time slots* a varios participantes en la red para la transmisión de sus mensajes, pero se van distribuyendo dinámicamente de forma automática a cada unidad de acuerdo a sus requisitos de transmisión en cada periodo de reasignación. El algoritmo de asignación de *time slots* reside en cada terminal, que puede calcular qué *time slots* debe usar en el siguiente periodo de reasignación de *slots*.

Cuando la suma de las demandas de bloques de *slots* de todos los terminales participantes en la red es menor que el número de bloques de *slots* disponible en el conjunto TSR, a cada terminal se le asigna un número de bloques de *slots* igual al demandado, y no existe ningún conflicto. Pero si es mayor, a cada terminal se le asigna un número de bloques de *slots* proporcional al número demandado, con lo que se va creando una cola de mensajes sin poder transmitirse en cada terminal, ya que no todos los mensajes podrán transmitirse en el siguiente periodo de reasignación y tendrán que esperar a los siguientes periodos.

El problema del algoritmo de reasignación actual es que permite conflictos en la asignación de bloques de *slots*. Además, intenta asignar el máximo número de bloques de *slots* a las unidades con una mayor prioridad según un orden, con lo que, a las unidades de menor prioridad les asigna un número bastante menor de bloques de *slots*. Esto quiere decir que la cola máxima de mensajes que no pueden ser transmitidos para algunos terminales es mucho mayor que para otros terminales.

En esta Tesis se estudia la mejora del algoritmo TSR (*Time Slot Reallocation*) de reasignación dinámica de *time slots* para una red Link-16, en las dos situaciones posibles: que la suma de las demandas de bloques de *slots* de todos los terminales participantes en la red sea mayor que el número de bloques de *slots* disponible, o que sea menor.

En el primer caso, se propone la utilización de diversas técnicas de Optimización Combinatoria. Con ello, se pretende eliminar los casos de conflicto en la asignación y minimizar el valor medio de la cola máxima de mensajes no transmitidos en cada terminal participante, es decir, que las colas de mensajes no transmitidos en todos los participantes en una red se mantengan lo más pequeñas posibles en todos los periodos de reasignación.

En el segundo caso, se propone la utilización de Técnicas Predictivas y Teoría de Juegos para asignar los bloques de *slots* que quedan sin asignar. Con ello se pretende minimizar el tiempo de transmisión de los mensajes para cada terminal participante en la red, desde que se demandan los *slots* hasta que se transmiten, ya que, al asignarle más bloques de *slots*, puede transmitir antes en el siguiente periodo de reasignación.

# ABSTRACT

The tactical military data communications networks have the purpose of the exchanging of tactical information among different military units, each one of these has a terminal for data communication, optimizing the joint operational military functions. Link-16 is at the moment the most advanced standard of these NATO networks. In this network, each terminal is assigned a pool of time slots to transmit their messages. There are three access modes to these time slots: dedicated access, contention access and Time Slot Reallocation access.

In Time Slot Reallocation (TSR) access mode, participants in the networks are assigned the same pool of time slots for the transmission of the messages, and these time slots are dynamically distributed to each transmitter according to its transmission requirements for each reallocation period. An algorithm within the terminals redistributes the pool of time slots in real time to meet these needs for the next reallocation period.

If the sum of all time slots blocks requested by all terminals in the network is less than the total number of time slots blocks available for TSR, each terminal is assigned the number of requested time slots blocks, and there is no conflict. But, if it exceeds the total number of time slots blocks available for TSR, each terminal is assigned a number of time slots blocks in proportion to the terminal´s request. So, a queue of messages that has not been transmitted is created in each terminal, because not all messages can be transmitted in the next reallocation period and will have to be transmitted in other periods.

The problem in the current assignment algorithm is that there are conflicts in the time slots blocks assignments. Moreover, it tries to assign as many of the time slots blocks as required to satisfy their requests to the terminals with a greater priority according to a priority order, so the terminals with a less priority are assigned a few time slots blocks. That is, the maximum queue of messages that can not be transmitted increase for some terminals.

In this Thesis, improved Time Slot Reallocation (TSR) algorithms of dynamic time slots assignment in Link-16 networks are studied for the two possible situations: the sum of all time slots blocks requested by all terminals in the network exceeds the total number of time slots blocks available for TSR, or it is less than the total number of time slots blocks available for TSR.

In the first case, the use of different Combinatorial Optimization Techniques is proposed. These new algorithms are focused on avoiding the conflicts in the time slots assignments and minimizing the maximum queue average of messages that can not be transmitted in each participant terminal, that is, the queues of messages that can not be transmitted in all participant terminals in the network keep the smallest as possible for all reallocation periods.

In the second case, the use of Predictive Techniques and Game Theory is proposed for the assignment of the slots blocks that not have been assigned. With these methods, the objective is to minimize the time for messages transmission, from the time slots blocks are requested to the messages are transmitted, since when terminals are assigned more time slots blocks, they can transmit before in the next reallocation period.

# Índice General

# Parte I

# Introducción y Revisión

# Capítulo 1 Introducción General

## 1.1 La red de comunicaciones Link-16.

Las redes de comunicaciones de datos tácticas militares responden a una necesidad militar operativa. El objetivo fundamental de este tipo de redes es el intercambio de información entre un gran número de unidades tácticas dispersas en un área, cada una de las cuales llevará un terminal para la transmisión de datos, optimizando las funciones militares operativas conjuntas [MIDSCO99]. Los tipos de información, a grandes rasgos, que estas redes intercambian se clasifican en tres grandes grupos:

a) Datos y voz para la realización de misiones.
b) Datos de navegación absoluta y relativa.
c) Datos de identificación.

Como es lógico, el sistema que soporte estas redes cumplirá los requisitos exigidos a cualquier sistema de armas moderno:

- Bajo tiempo de respuesta.
- Uso de técnicas de encriptado para la seguridad de los datos.
- Elevada protección *anti-jamming*.
- Muy alta resistencia a las perturbaciones y contramedidas electrónicas.
- Arquitectura distribuida: esto permite la ausencia de nodos críticos.
- Alta capacidad para la emisión de datos.
- Interoperatividad con el resto de sistemas automáticos de tratamiento de datos.

El ejemplo más avanzado de este tipo de redes de comunicaciones tácticas militares utiliza para la transmisión de datos el protocolo Link-16. Esta red utiliza una arquitectura de Acceso Múltiple por División en el Tiempo (TDMA) [Mangat95], opera en la banda de frecuencias L (960 – 1215 MHz), y utiliza técnicas de codificación por secuencia directa, ampliación del espectro, salto de frecuencias (*frecuency hopping*), detección y corrección de errores y criptografía.

Estas redes pueden operar en dos modos: canal único y multicanal [Mangat97]. El modo de canal único puede considerarse como un único "bus de datos en el cielo". Cada participante, en dicha red única, tiene asignado un número de intervalos de tiempo (*time slots*) en el que hace sus transmisiones. Esto se puede ver en la Fig.1.1. Estas asignaciones se realizan cargando unos parámetros de inicialización en el terminal antes de que éste empiece a operar. Estos parámetros son los que ordenan a cada terminal que transmita en tal o cual instante de tiempo.

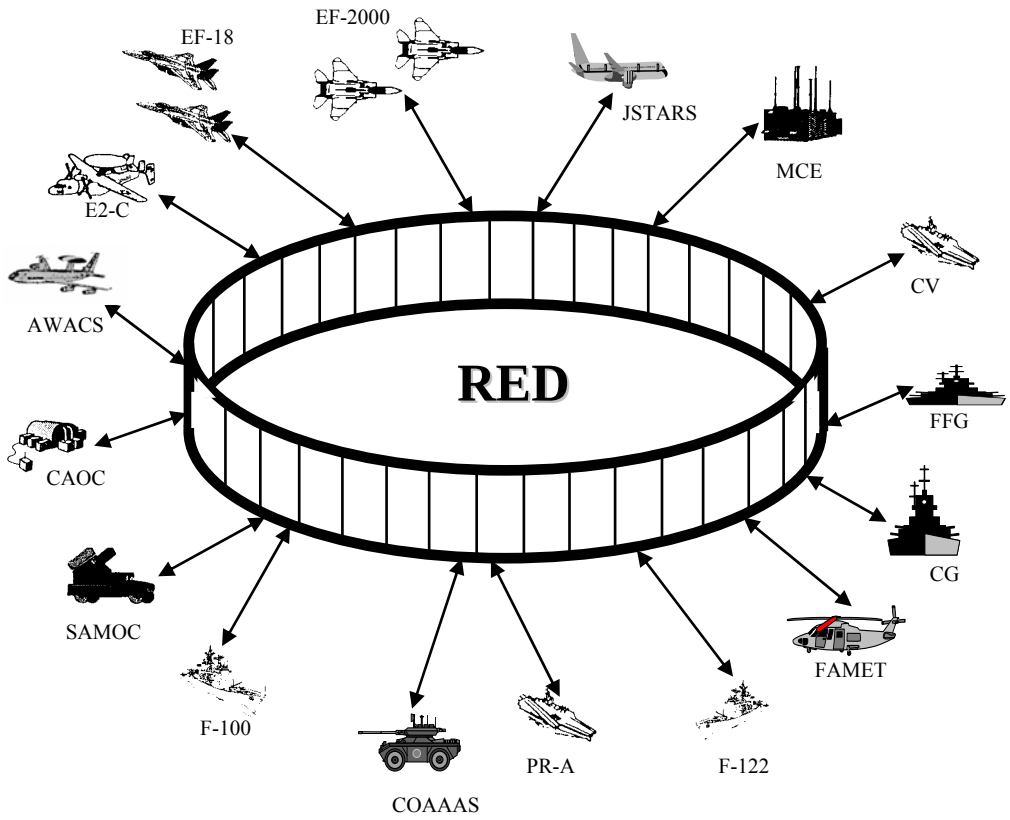



Fig. 1.1: Representación de una red Link-16

Dentro de cada intervalo de tiempo se transmiten una serie de pulsos que son los portadores de la información. La cantidad de *bits* de información que se transmiten en cada intervalo de tiempo depende del formato del mensaje escogido. Para encriptar todos los mensajes antes de ser transmitidos, cada terminal utiliza unas claves criptográficas, llamadas criptovariables. Estas criptovariables se utilizan para seleccionar la frecuencia de transmisión de cada pulso, de tal forma que la secuencia de frecuencias elegida aparezca como aleatoria para cualquier terminal que no posea las criptovariables [Logicon94]. Por lo tanto, para que un terminal pueda recibir las transmisiones de otro, deberá conocer las claves criptográficas utilizadas por el transmisor. Las asignaciones de recepción también se realizarán mediante los parámetros de inicialización. Con el cambio de frecuencias de un pulso al siguiente se consigue evitar la acción de un *jammer* de banda estrecha, es decir, el de un transmisor sintonizado exactamente a la frecuencia de transmisión.

Esta característica de cambio de frecuencias puede utilizarse para realizar comunicaciones simultáneas sin interferencia entre ellas, lo que constituye el modo de operación multicanal. Un terminal puede transmitir una serie de pulsos en los intervalos de tiempo que se le han asignado en una serie de frecuencias determinada, mientras que otro terminal transmite otra serie de pulsos en los mismos intervalos de tiempo, pero con una secuencia de frecuencias diferentes. Cada secuencia de frecuencias puede ser considerada como una red o canal. Este modo de operación se puede observar en la Fig.1.2. La red Link-16 contempla 127 redes (0-126) disponibles para los terminales [Barto91]. Cualquier terminal podrá transmitir o recibir en cualquier red, durante cualquier intervalo de tiempo. El salto de frecuencias depende del número de red y de la variable criptográfica utilizada. Estas frecuencias se escogen entre 51 frecuencias distintas, separadas 3 MHz una de otra, en la banda L (960-1215 MHz).

Los terminales están diseñados de tal forma que puedan separar sus mensajes en categorías diferentes, según el tipo de información o el destino del mensaje. Esta característica va unida a la asignación de intervalos de tiempo, es decir, un grupo de intervalos de tiempo se asignará para transmitir una determinada categoría de mensajes (o grupo de participación), basado en unos requisitos operacionales.

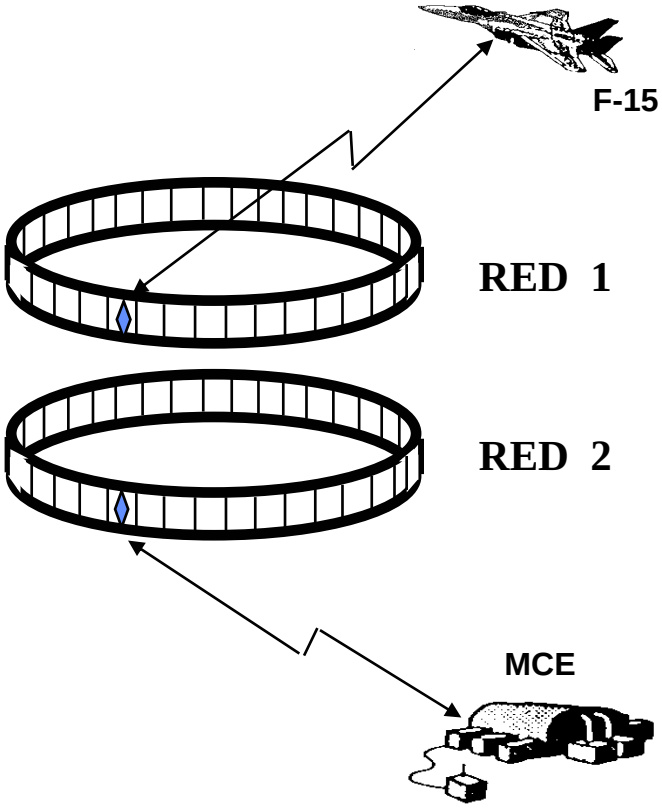



Fig.1.2: Modo de operación multicanal en la red Link-16

## 1.2 Arquitectura temporal de la red Link-16.

Desde el punto de vista temporal, el sistema divide las 24 horas del día en periodos denominados épocas (*epochs*) [MIDSCO99]. Cada día tiene 112,5 *epoch* de 12,8 minutos de duración. Cada *epoch* tiene 64 tramas (*frames*) de 12 segundos de duración, que a su vez se dividen en 1536 *time slots* de 7,8125 milisegundos (mseg).

El *time slot* es la unidad básica de la red Link-16. A cada unidad participante en la red se le asigna un conjunto de *time slots* en los que transmite o recibe información. Los 98304 *time slots* de una *epoch* se agrupan en tres grupos de *time slots* iguales: *Set* A, B, C, cada uno de los cuales contiene 32768 *time slots*. Dentro de una *epoch* los *time slots* de cada *set* están entrelazados con los de los otros *sets* en la siguiente secuencia:

A-0, B-0, C-0

A-1, B-1, C-1

A-2, B-2, C-2

.....

A-32767, B-32767, C-32767

La Fig.1.3 muestra la estructura temporal descrita anteriormente.

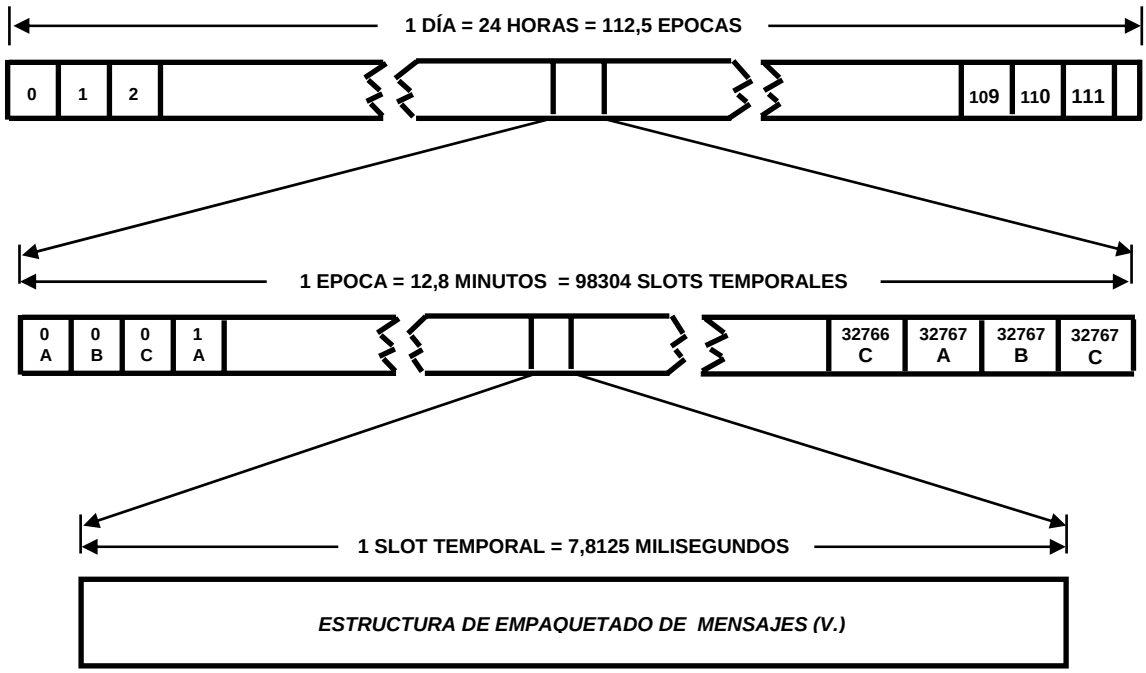



Fig.1.3: Estructura temporal en la red Link-16

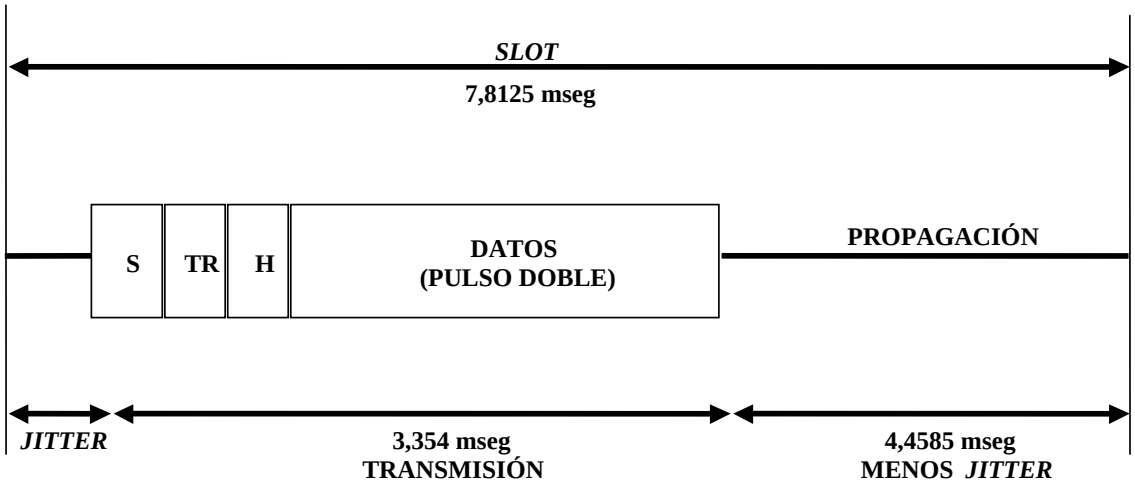
Para el diseño de una red Link-16, se usa la trama (*frame*) de 12 segundos de duración, que contiene 1536 *time slots*. Un *frame* también se divide en tres grupos de *time slots* iguales y entrelazados: *Sets* A, B, C. Cada uno de ellos contiene 512 *time slots*.

Los *time slots* se asignan a cada unidad participante en bloques llamados *Time Slot Blocks* (TSB). En estos bloques, la unidad transmite o recibe información. Un TSB se identifica por los siguientes parámetros: *Set* al que pertenece, *Index Number* y *Recurrence Rate Number*.

Todos los *time slots* de un TSB pertenecen al mismo *Set*, y están equiespaciados en el tiempo. Como los *Sets* están entrelazados entre si, los TSBs también lo están. El *Index Number* indica el primer *time slot* del TSB. Su rango es 0 – 32767. Finalmente, el *Recurrence Rate Number* (RRN) indica el tamaño del TSB. El tamaño de un TSB se expresa como una potencia de 2, y es $2^{RRN}$ *time slots* por época *(epoch)*. Su rango es 0 – 15 (desde $2^0$, 1 *time slot* por época, hasta $2^{15}$, 32768 *time slots* (un *set* completo) por época). El RRN determina con qué frecuencia ocurre el *time slot* durante una época. Por ejemplo, un TSB con un RRN de valor 6 ($2^6$ = 64) tiene un total de 64 *time slots* equiespaciados por época, lo que equivale a 1 *time slot* por *frame*.

### 1.2.1 Estructura del *time slot.*

La Fig.1.4 muestra la estructura de datos del *time slot*. Cada *time slot* comienza con un tiempo muerto llamado *jitter*, en el que no se transmiten datos. La duración del *jitter* varía de forma pseudoaleatoria, dependiendo de la criptovariable TSEC (seguridad de la transmisión). El *jitter* contribuye a hacer la señal más resistente a la perturbación, ya que impide al posible perturbador conocer el momento de comienzo de la señal.




**S = SINCRONIZACIÓN**
**TR = REFINAMIENTO DE TIEMPO**
**H = CABECERA (*HEADER*)**

Fig.1.4: Estructura de datos del *time slot*

A este tiempo muerto le sigue un espacio ocupado por el mensaje (sincronización, cabecera y datos), donde se transmite una secuencia de pulsos que contienen la información en paquetes de símbolos codificados [Logicon94]. Cada pulso representa 5 *bits* y tiene una duración de 13 microsegundos (μseg): la portadora es modulada durante los primeros 6,4 μseg, que van seguidos por 6,6 μseg de tiempo muerto. La cabecera del mensaje especifica el tipo de formato del mensaje que sigue, si está o no codificado y el tipo de empaquetado. Está formada por una serie de 35 *bits* que son codificados con el algoritmo *Reed-Solomon* (R-S) para detección y corrección de errores, transformándose en 80 *bits*. Estos se agrupan en grupos de 5 *bits* para formar 16 símbolos codificados.

El *time slot* finaliza con un tiempo muerto de propagación que permite a la señal un alcance normal de 300 millas o un alcance extendido de 500 millas.

Finalmente, la forma de onda se genera mediante una modulación por desplazamiento continuo de fase (CPSM) de la frecuencia portadora, a una tasa de 5 Mbps, utilizando para ello como señal moduladora las secuencias de 32 *bits* correspondientes a los símbolos de transmisión.

### 1.2.2 Tipos de empaquetado.

En cada *time slot*, la información se puede transmitir en bloques de datos con pulso sencillo (SP) o doble (DP). Con pulso sencillo, la información se transmite una sola vez, mientras que con pulso doble, la información se repite a frecuencia distinta. La estructura de pulso doble proporciona redundancia para aumentar la probabilidad de recepción y aumenta la capacidad *antijamming* de la red. Hay cuatro estructuras de empaquetado, que se muestran en la Fig.1.5:

- *Standard Double Pulse* (STD).
- *Packed-2 Single Pulse* (P2SP).
- *Packed-2 Double Pulse* (P2DP).
- *Packed-4 Single Pulse* (P4SP).

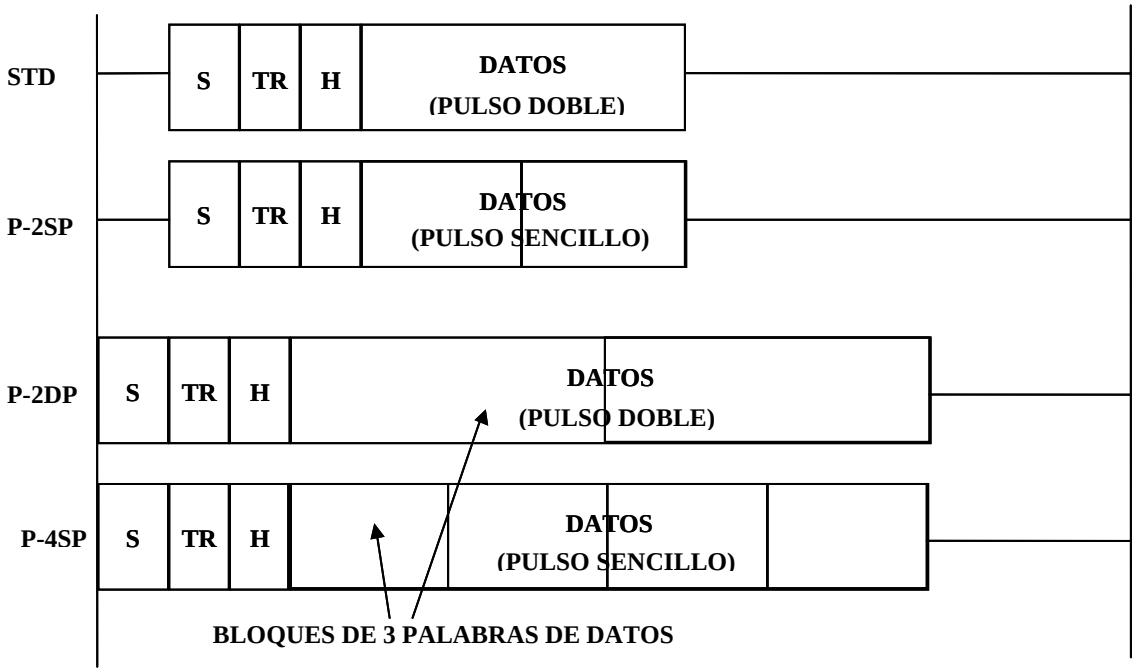



Fig.1.5: Estructuras de empaquetado de datos

El tipo de empaquetado STD consigue la máxima protección de los datos. Transmite un bloque de datos con pulso doble, es decir, a dos frecuencias distintas de forma consecutiva. El tipo de empaquetado P2SP es menos resistente a las contramedidas que el anterior debido a que los datos se transmiten una sola vez, pero con esta estructura se transmiten dos bloques de datos con pulso sencillo. Con el tipo de empaquetado P2DP se transmiten también dos bloques de datos, pero con pulso doble. Con esto se consigue una redundancia de datos similar al tipo STD y una capacidad de transmisión de datos igual al tipo P2SP. Para ello se elimina el *jitter*, con lo que las transmisiones podrían ser perturbadas por medio de contramedidas sincronizadas con la base de tiempos del sistema. Finalmente, el tipo de empaquetado P4SP es el que mas capacidad de transmisión de datos consigue, cuatro bloques de datos con pulso sencillo, pero es la menos resistente a las contramedidas. Para ello, como en el tipo P2DP, se elimina el *jitter*.

## 1.3 Estructura de los mensajes.

Los datos se agrupan en bloques de tres palabras, de 75 *bits* (155 con la redundancia) cada una. Según el tipo de empaquetado antes mencionado, en un *time slot* pueden transmitirse: un bloque de tres palabras (*Standard*), dos bloques (*Packed*-2), o cuatro bloques (*Packed*-4). En caso de que no sea múltiplo de tres el número de palabras a transmitir, se introducen palabras de relleno.

Link-16 dispone de cuatro tipos de mensajes distintos:

- Formato fijo. Serie J.
- Formato variable.
- Texto libre.
- RTT.

### 1.3.1 Formato fijo. Serie J.

Se utilizan para el intercambio de información táctica y de mando. Están formados por una palabra inicial, que puede ir seguida de una ó más palabras de extensión y una o más palabras de continuación (en total, de 1 a 8 palabras).

Cada palabra consta de 75 *bits*: 70 de datos, 4 de paridad y 1 vacía (de reserva). La palabra pasa a contar con 155 *bits* tras aplicarle la codificación *Reed-Solomon* (R-S) para detección y corrección de errores.

Los mensajes se distribuyen por contenidos de acuerdo al etiquetado especificado por el estándar Link-16 [Stanag16].

### 1.3.2 Formato variable.

Para compatibilidad con redes de comunicaciones tácticas militares Link-11 B, más antiguo que el sistema Link-16, utilizan la misma estructura de palabras, pero los formatos no se ajustan a ella (pueden ser más pequeñas o mayores que una palabra).

### 1.3.3 Texto libre.

No tienen un formato específico, se usan para transmitir voz, vídeo o texto. Utiliza 75 *bits* por palabra. Un mensaje de 3 palabras contiene 225 *bits*. Estos mensajes pueden ir o no codificados con técnicas de detección y corrección de errores R-S. Si se utiliza la codificación R-S, los 225 *bits* de información se transforman en 465 *bits* para la transmisión, siendo los *bits* adicionales la redundancia del código. No obstante, en este formato solo se utilizan 450 de los 465 *bits*, con objeto de acomodar la tasa binaria a lo necesario para la transmisión de voz (2400 bps, 4800 bps, etc.).

### 1.3.4 RTT (*Round Trip Timing*).

Son mensajes de pregunta y respuesta para sincronizar un terminal con la red. Son similares a las cabeceras del resto de mensajes, pero sin que les sigan bloques de datos. Además, no llevan *jitter*.

## 1.4 Modos de acceso.

Como se ha comentado en el apartado 1.2 de este capítulo, a cada participante en una red de comunicación de datos tácticos militares se le asigna un grupo de intervalos de tiempo (*time slots*) para transmitir un determinado tipo de mensajes (o grupo de participación). Existen tres modos de acceso a esos *time slots* [Mangat95, Mangat97]: acceso dedicado, acceso por contienda y acceso por reasignación de *time slots* (*Time Slot Reallocation*).

### 1.4.1 Acceso dedicado.

En este modo de acceso se asignan *time slots* de transmisión a cada participante en una red, y en esos *time slots* solamente puede transmitir ese participante. Esto significa que no existen conflictos en la transmisión de los mensajes, y se garantiza su recepción. Las transacciones en este modo de transmisión no requieren el análisis de probabilidades de recepción. Basta con asignar el número apropiado de *slots* a cada

unidad para poder transmitir todos sus mensajes. Con este modo de acceso se transmiten, por ejemplo, los mensajes de gestión de red, ya que, al ser mensajes críticos, se necesita garantizar su recepción.

Con el modo de acceso dedicado, el número de *time slots* necesarios crece linealmente con el número de unidades en la red, ya que cada unidad tiene asignados sus propios *slots* de transmisión, que ninguna unidad puede utilizar. Esto hace que se necesite un proceso de "gestión": cuando una nueva unidad quiere transmitir, no puede utilizar ningún *slot* asignado ya a otra unidad. Debe ser inicializada con aquellos *slots* que no están siendo utilizados.

Existe otro modo de acceso dedicado llamado acceso dedicado con reuso de *slots*, que permite a más de una unidad transmitir con acceso dedicado en los mismos *time slots*. Se usa cuando las unidades se encuentran en zonas demasiado alejadas unas de otras que no afectan unas a otras, con lo que pueden transmitir en los mismos *slots*.

### 1.4.2 Acceso por contienda.

En este modo de acceso se asigna un mismo conjunto de *time slots* a varios participantes en la red para la transmisión de sus mensajes, y cada participante tiene asignada una frecuencia de acceso a esos *time slots*. Cada unidad participante selecciona los *slots* de ese conjunto de manera que su selección es aleatoria respecto a la de los otros terminales. Esto significa que alguna vez varios terminales pueden seleccionar el mismo *slot* para transmitir sus mensajes. En este caso, los terminales receptores escucharán el transmisor más cercano físicamente a ellos. Con este modo de acceso se transmiten, por ejemplo, los mensajes de posición de las unidades, ya que, en el caso de transmisión en el mismo *slot*, recibe la posición del transmisor más cercano, que es lo que más puede interesar al terminal.

Debido a la selección aleatoria de los *slots* por parte de las unidades, no se puede garantizar la recepción del mensaje. Esto requiere el uso de probabilidades estadísticas de recepción, que dependerá del número de unidades que transmitan en estos *slots*, el

número de *slots* disponibles y el número de *slots* que seleccionen las unidades para transmitir.

En un conjunto de *slots* en modo de acceso por contienda no se necesita ningún proceso de "gestión": se pueden añadir nuevas unidades a ese grupo de *slots* sin preocuparse de los *slots* que va a utilizar para su transmisión, ya que todas las unidades utilizan el mismo conjunto de *slots*. En este sentido, es un modo de acceso más flexible que el dedicado. Evidentemente, cuanto mayor sea el número de unidades menor será la probabilidad de recepción de los mensajes, y mayor será la incertidumbre en la transmisión.

### 1.4.3 *Time Slot Reallocation* (TSR).

Es un modo de acceso intermedio entre los dos anteriores. Como en el acceso por contienda, en este modo de acceso se asigna un mismo conjunto de *time slots* a varios participantes en la red para la transmisión de sus mensajes [Barto91], pero se van distribuyendo dinámicamente de forma automática a cada unidad de acuerdo a sus requisitos de transmisión en cada momento, es decir, cada participante recibe un número de *slots* de ese conjunto según sean sus necesidades en cada momento. Cada unidad tiene modo de acceso dedicado a los *slots* que recibe. Este modo de acceso se usa cuando las necesidades cambian rápidamente de un instante a otro. Por ejemplo, para los mensajes de vigilancia.

TSR combina la flexibilidad del acceso por contienda con la recepción garantizada del acceso dedicado. Naturalmente hay que pagar un precio por esto: se necesitan algunos *slots* para transmitir las demandas de *slots* de todas las unidades participantes en cada momento. Estos *slots* se toman del conjunto de *slots* asignado para TSR. Después de intercambiar estos mensajes, cada terminal puede calcular qué *time slots* debe usar en el siguiente periodo TSR. Ya que en cada terminal reside el mismo algoritmo de asignación de *time slots*, las unidades participantes seleccionan los *slots* en los que transmiten sin conflictos (no existe ninguna unidad controladora, cada participante calcula de forma independiente su propia asignación de *time slots*). Al ser el algoritmo TSR el que gestiona los *slots* del conjunto asignado para TSR, no existe

ningún tipo de "gestión humana". Nuevos participantes pueden entrar a transmitir en ese conjunto de *slots*, monitorizar los *slots* de los demás participantes y empezar a transmitir en sus propios *slots*.

El modo de acceso TSR descrito hasta ahora es el modo descentralizado. Pero, debido a que el conocimiento por parte de una unidad de la demanda de *slots* del resto de unidades puede ser incompleto (una unidad puede que no reciba todos los mensajes de demanda de *slots* del resto de unidades), puede haber errores en la asignación de *slots* que ocasionen más colisiones de las previstas. Por eso en tiempo de paz se emplea otro modo de acceso TSR, el centralizado. Este modo opera con un terminal funcionando como *net manager*, que controla los mensajes que recibe de todos los participantes y realiza el proceso de reasignación de *slots* para todas las unidades. Un mensaje de comando asigna a cada unidad su asignación de *slots*.

En el modo de acceso TSR el tiempo se divide en periodos, llamados *reallocation period* [MIDSCO99], de duración variable entre 6 y 48 segundos. En cada periodo, cada terminal se asignará mediante un algoritmo de selección un número de bloques de *slots* para la transmisión de sus mensajes en el siguiente periodo. Cuando la suma de las demandas de bloques de *slots* de todos los terminales participantes en la red es menor que el número de bloques de *slots* disponible en el conjunto TSR, a cada terminal se le asigna un número de bloques de *slots* igual al demandado, y no existe ningún conflicto. Pero si la suma de las demandas de bloques de *slots* de todos los terminales es mayor que el número de bloques de *slots* disponible en el conjunto TSR, a cada terminal se le asigna un número de bloques de *slots* proporcional al número demandado, con lo que se va creando una cola de mensajes sin poder transmitirse en cada terminal, ya que no todos los mensajes podrán transmitirse en el siguiente *reallocation period* y tendrán que esperar a los siguientes periodos. Esto hace que el tiempo que transcurre entre que se demanda la transmisión de un mensaje y se transmite ese mensaje puede crecer de forma importante.

En el algoritmo de asignación actual, las unidades participantes en el conjunto de *slots* asignado a este modo de acceso se ordenan según un orden de prioridad basado en un parámetro llamado *receive count*. El *receive count* de cada terminal participante es el número de mensajes de demanda de *slots* que recibe del resto de participantes en cada

*reallocation period* [MIDSCO99]. El terminal con menor *receive count* tiene mayor prioridad. Los terminales cuyo *receive count* es cero se llaman terminales sordos. No reciben ningún mensaje de demanda de *slots*, por lo que no tienen conocimiento alguno de la demanda del resto de terminales. Por eso, el algoritmo asigna a los terminales sordos todos los bloques de *slots* que solicitan, permitiendo conflictos entre ellos. Y luego se asignan el resto de bloques a los demás terminales según el orden de prioridad. En cada terminal, el algoritmo intenta asignar todo lo que solicitan los terminales con menor *receive count* que el del propio terminal, después lo que solicitan los terminales con igual *receive count* que el del propio terminal y, finalmente, lo que solicitan los terminales con mayor *receive count* que el del propio terminal, hasta que se complete el conjunto de *slots*. A los participantes que faltan se les asigna un bloque de *slots*. El algoritmo de asignación completo se explica de forma detallada en el Apéndice A de esta tesis. Así pues, existen casos con conflictos en la asignación. Además, con el orden de prioridad, el número de bloques de *slots* que se deja de asignar a los últimos terminales según el orden de prioridad es mucho mayor que para los primeros, con lo cual la cola máxima de mensajes que no pueden ser transmitidos para algunos terminales crece de forma innecesaria.

## 1.5 Grupos de participación.

Los grupos de participación (NPG, *Network Participation Groups*) definen una estructura lógica para la transmisión y recepción de mensajes Link-16. Establecen el interés de un participante de la red por recibir los mensajes de una determinada área de funcionalidad. No es necesario que todos los participantes reciban toda la información disponible en la red. En el proceso de diseño de una red Link-16, a cada unidad participante en la red se le asignará los bloques de *time slots* necesarios para cada uno de los NPGs en los que transmita o reciba información, dependiendo de sus necesidades operativas específicas.

Además del grupo para información de posición e identificación de fuerzas amigas, al que pertenecen todas las unidades, los grupos funcionales pueden incluir:

- Vigilancia.
- Selección de blanco entre aviones caza ("fighter-to-fighter").
- Control aéreo.
- Coordinación e informes de Guerra Electrónica.
- Gestión de la misión y control de armas de la fuerza.
- Dos canales de voz seguros.
- Acceso inicial a la red.
- Control / distribución del tiempo de red.

Los NPGs definidos actualmente por la OTAN [ADatP97] son los siguientes:

| NPG núm | Nombre NPG | NPG núm | Nombre NPG |
|---|---|---|---|
| **1** | Initial Entry | **13** | Voice Group B |
| **2** | RTT A | **14** | Indirect PPLI |
| **3** | RTT B | **19** | Fighter-to-fighter targetting |
| **4** | Network Management | **21** | Engagement Coordination |
| **5** | PPLI & Status, Group A | **22** | Composite A |
| **6** | PPLI & Status, Group B | **23** | Composite B |
| **7** | Surveillance | **27** | Joint Net PPLI |
| **8** | Miss.Manag./Weap.Coord.&Manag. | **29** | Residual Message |
| **9** | Air Control | **30** | IJMS Position & Status |
| **10** | Electronic Warfare | **31** | IJMS Message |
| **12** | Voice Group A | | |

A continuación, se describe brevemente cada grupo de participación:

- NPG 1: *Initial Entry* (Entrada inicial).

  Apoya la sincronización gruesa y la entrada inicial en red. La unidad designada como referencia de la hora en la red (NTR, *Network Time Reference*) transmite de forma periódica los mensajes de entrada en red estándar en este NPG, y es utilizado por las demás unidades para sincronización de la hora. Los mensajes de entrada en red son transmitidos por aquellas unidades designadas como IEJU (*Initial Entry Joint Unit*) y por las unidades relés. Normalmente los mensajes de entrada en red se transmiten en el primer *time slot* de cada *frame* y, puesto que efectúa la sincronización y la entrada en red, es necesario que todas las unidades participen en este grupo.

- NPG 2: RTT (*Round Trip Timing*) A (dedicado).
  Para la sincronización fina se generan automáticamente en cada terminal unos mensajes especiales *Round Trip Timing* que se transmiten por este NPG. También facilita la incorporación posterior de nuevas unidades a la red, y facilita la navegación relativa. Durante la sincronización se mandan los mensajes RTT hasta tres veces cada *frame*. Una vez sincronizada, se envía una vez cada minuto. De no estar incluido éste NPG, los mensajes RTT se distribuyen en los NPGs 5 o 6.

- NPG 3: RTT (*Round Trip Timing*) B (contienda).
  Lleva a cabo la misma función que el NPG 2, pero los *time slots* son compartidos por un grupo de unidades usando el método de acceso por contienda, en el cual todas las unidades pueden transmitir.

- NPG 4: *Network Managenent* (Gestión de red).
  Este grupo permite la redistribución de la capacidad de la red conforme a necesidades. Cada unidad de mando y control (C2) tiene normalmente cierta capacidad de administración (aunque todos los terminales son capaces de procesar los mensajes). Por ejemplo, una unidad controladora de una aeronave podría cambiar la asignación de *time slots* de respuesta de la aeronave para permitirle trabajar en el NPG de Control Aéreo.

- NPG 5: PPLI (*Precise Participant Location and Identification*) *and Status*, *Pool* A.
  Se usa por unidades no C2 (normalmente aviones de combate) en unión del NPG 6. Utilizando los *time slots* de ambos grupos, las aeronaves son capaces de trasmitir su posición con un periodo de refresco mayor. Los barcos no necesitan actualizar su posición tantas veces como las aeronaves. La parte final de los mensajes lleva información sobre el estado de armas y combustible remanente, que se trasmite automáticamente por cada terminal que pertenece a este grupo.

- NPG 6: PPLI (*Precise Participant Location and Identification*) *and Status*, *Pool* B.
  Se usa por todas las unidades (C2 y no C2) para identificación, sincronización y navegación relativa. Además de la información correspondiente a la identificación y posición exacta, los mensajes PPLI incluyen los distintivos numéricos de red y

control aéreo que usa cada plataforma. Por defecto, cuando no ha sido definido un NPG RTT, el terminal envía los mensajes RTT en este grupo.

- NPG 7: *Surveillance* (Vigilancia).
  La vigilancia consiste en la búsqueda, detección, identificación y seguimiento de contactos. Los contactos aéreos, de superficie y submarinos, los emplazamientos terrestres, puntos de referencia, puntos de ASW, demoras acústicas y demoras de Guerra Electrónica (EW) se intercambian todos en este NPG. Todos los buques y aeronaves de alerta previa participan en este grupo. Los aviones de combates normalmente no pueden monitorizar ni procesar la información de este grupo, la situación aérea se la reencamina su controlador. Sin embargo, las aeronaves pueden hacer de relé de este grupo al inicializarse el equipo.

- NPG 8: *Mission Management* (Gestión de la misión) / *Weapons Coordination & Management* (Gestión y Coordinación de armas).
  Se usa para que la unidad designada pueda coordinar las armas de fuerza, además de facilitar el estado de las armas. La información de este NPG se ha dividido en dos NPGs, en éste y el 21, dejando éste para información automática del estado de las armas y reservando el NPG 21 exclusivamente para órdenes de enfrentamiento.

- NPG 9: *Air Control* (Control Aéreo).
  Este grupo facilita que las unidades C2 controlen a las no C2. Está divido en dos componentes, mensajes *uplink* y mensajes *backlink*. La unidad controladora envía la misión asignada, vectores e informes de blanco al avión en los *time slot*s asignados para el *uplink*, por lo que el avión recibe una información procesada que correlaciona con sus sensores. Para el *backlink* (o *downlink*), los aviones transmiten sus propios contactos radar, respuestas a las órdenes y estado de plataforma a su controlador.

- NPG 10: *Electronic Warfare* (Guerra Electrónica).
  Este grupo permite la distribución de órdenes e información de datos de Guerra Electrónica (EW) entre las unidades que tengan esta capacidad. A menudo se hace conjunto con el NPG 5 (PPLI de aviones), ya que estos no tienen capacidad EW.

- NPGs 12 y 13: *Voice Group A* (Voz Grupo A) y *Voice Group B* (Voz Grupo B).
  Facilita dos canales de voz digitalizados y seguros a todas las unidades.

- NPG 14: *Indirect PPLI* (PPLI Indirecto).
  Facilita las operaciones multilink, asignando *time slots* en los que las unidades designadas como FJU (*Forwarding Joint Unit*) retransmiten los mensajes PPLIs, que llevan la información de identificación y posición, a las unidades de Link-11 que no están en la red de Link-16.

- NPG 19: *Fighter to Fighter* (Caza-Caza).
  En este grupo, unidades no C2, como los aviones de combate, intercambian información de blancos de sus sensores. Debido a que los terminales en los aviones no están preparados para mantener activos simultáneamente este grupo y el de Control Aéreo (NPG 9), para mantener las comunicaciones entre grupos de aviones y controlador se reservan *time slots* de este grupo para el controlador.

- NPG 21: *Engagement Coordination* (Coordinación de enganche).
  Para evitar conflictos entre mensajes automáticos de estado de armas y órdenes de enfrentamiento críticas, se separa en este grupo la Coordinación de Armas del grupo NPG 8. Solo los mandos transmiten en este NPG.

- NPG 27: *Joint Net PPLI* (PPLI Conjunto).
  En este NPG se puede intercambiar información de posición e identificación durante operaciones conjuntas.

- NPG 29: *Residual Message* (Mensaje Residual).
  Es un grupo especial para asegurar la transmisión de mensajes que, considerándose necesarios, no han sido asignados a ningún otro grupo. Así ocurre cuando determinados mensajes son normalmente asociados a un NPG que no ha sido incluido en la red. Otro ejemplo son los mensajes J28, que permiten intercambiar mensajes de texto.

- NPG 30: *IJMS Position & Status* (Posición y Estado IJMS).
  En aquellas unidades cuyo terminal pueda operar conjuntamente en una red Link 16 y en una IJMS (*Interim JTIDS Message Specification*) y puedan trabajar en las dos redes, retransmitirán en este NPG el mensaje de posición y estado IJMS. Esto tan solo lo llevan a cabo determinados aviones de la Fuerza Aérea Norteamericana.

- NPG 31: *IJMS Messages* (Mensajes IJMS).
  El resto de mensajes y canal de voz de la red IJMS se trasmiten en este grupo. Aunque los terminales de las unidades no sean capaces de trabajar con unidades de una red IJMS, caso de existir ésta se comparte la misma estructura de red, haciendo así las dos redes compatibles sin interferencias.

## 1.6 Uso de redes múltiples.

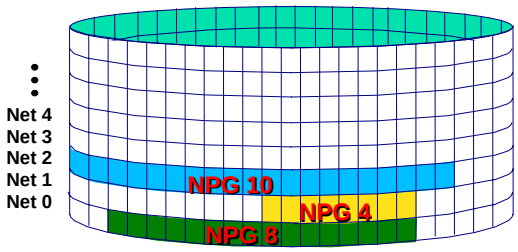
Como se ha comentado en el apartado 1.1 de este capítulo, la red Link-16 puede operar en dos modos: canal único, o una sola red, y multicanal, o varias redes simultáneamente. La estructura de transmisión de la red Link-16 permite la definición de 127 redes diferentes. El número de la red, el cifrado de la señal (criptovariable TSEC) y el número de *time slot*, determinan el patrón de salto de frecuencia de la portadora en cada red. Los diferentes patrones de salto de frecuencia son los que mantienen la separación de redes y permiten que diversas redes puedan operar a la vez.

Se pueden hacer diferentes redes simplemente especificando distinto número de red para un mismo NPG, sin modificar las criptovariables TSEC (seguridad de la transmisión) y MSEC (seguridad del mensaje). Si las criptovariables TSEC y MSEC se cambian se pueden hacer varios tipos de redes múltiples, incluyendo relés ciegos. Los tipos más comunes son las redes apiladas y las cripto-red.

### 1.6.1 Redes múltiples (*multinetting*).

Las redes múltiples se pueden establecer simplemente especificando diferentes números de red. Aunque son 127 las posibles redes, estudios estadísticos muestran que

para no existir interferencias entre los diferentes patrones de salto de frecuencia no se deben establecer más de 20. La Fig.1.6 muestra la estructura de estas redes.




| Time Slot Block | NPG | Net # | MSEC | TSEC |
|---|---|---|---|---|
| A - 8 - 10 | 8 | 0 | 1 | 1 |
| A - 40 - 9 | 4 | 1 | 1 | 1 |
| A - 0 - 12 | 10 | 2 | 2 | 1 |

Fig.1.6: *Multinetting*

Cada bloque de *time slots* se asigna a los participantes para un NPG distinto en cada red, teniendo en cuenta que cada participante solo puede transmitir o recibir en una de ellas en cada *time slot*. Se utiliza cuando un participante no necesita escuchar todos los tipos de información que se transmite en los mismos *time slots*, solo se le asigna el bloque de *time slots* en una red.

### 1.6.2 Redes apiladas (*Stacked Nets*).

Se crean asignando los mismos bloques de *time slots* para el mismo NPG con la misma criptovariable TSEC en todas las redes. Se definen con el parámetro *Net Number* a un valor de 127. Previo a la operación, se determina la red en la que transmitirá cada grupo de participantes. Estas redes son usadas para que varios grupos de participantes transmitan simultáneamente en distintas redes. Los NPGs de Voz y Control Aéreo utilizan este tipo de redes. Por ejemplo, para control de aeronaves se asignan los mismos bloques de *time slots* para el NPG 9 (Control Aéreo), y cada grupo de unidad controladora y sus unidades controladas utilizan una red distinta. La Fig.1.7 muestra la estructura de estas redes.

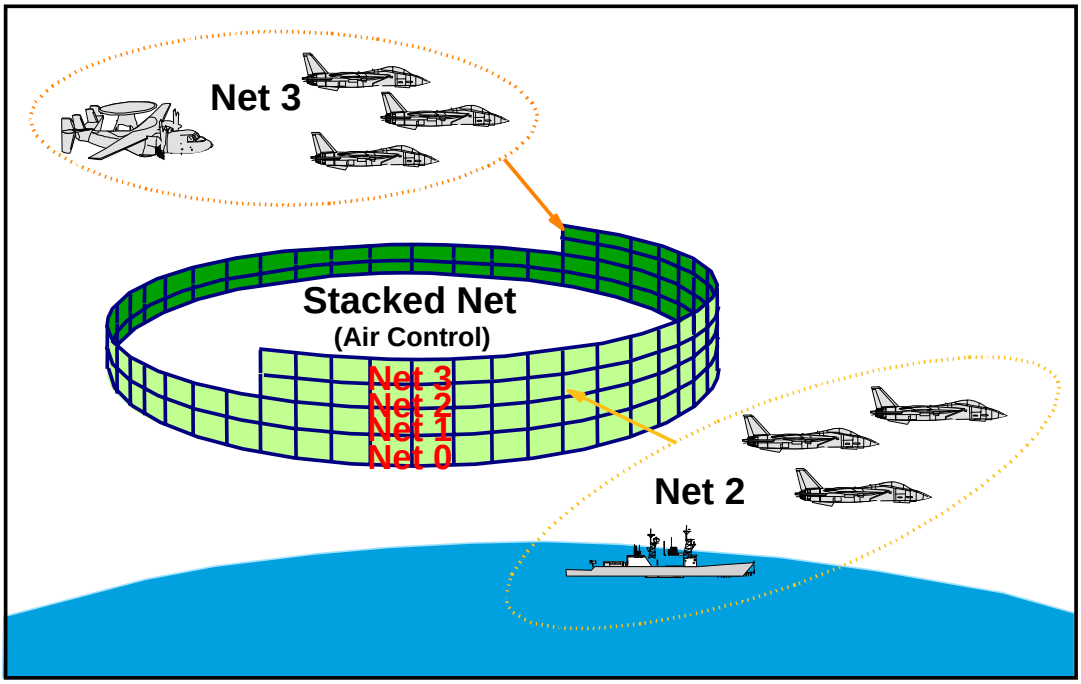



Fig.1.7: *Stacked Nets*

### 1.6.3 Cripto-redes (*Crypto-Nets*).

El aislamiento entre redes o entre usuarios de una red se realiza introduciendo en el terminal distintas criptovariables. Así, solo aquellos que tengan las mismas criptovariables podrán intercambiar información. Las cripto-redes se crean asignando diferentes criptovariables TSEC y/o MSEC.

Si la criptovariable TSEC es la misma y la MSEC es distinta, usuarios no autorizados pueden recibir la señal, llevar a cabo la corrección de errores y retransmitirla, pero no la pueden descifrar. Esta es la configuración que se usa para los relés ciegos. Si ambas criptovariables son distintas, el aislamiento entre unidades es total.

## 1.7 Proceso de Gestión de las redes Link-16.

El propósito de la Gestión de redes Link-16 es asegurar que los participantes puedan explotar la inherente flexibilidad y capacidad de los terminales y extraerles el máximo beneficio operativo. Para conseguir el máximo rendimiento de una red Link-16, es necesario facilitar a todos los participantes los parámetros que les permitan tener la misma definición de las características de la red, el uso coordinado de *time slots*, y el uso coordinado de las variables criptográficas.

La Gestión de red debe de tener los siguientes objetivos:

- La red debe servir de apoyo a las Operaciones Tácticas planeadas.
- La red debe ser común para tiempos de paz y de guerra.
- La red debe ser robusta y sin nodos.
- La red debe contener las mínimas restricciones para el alistamiento operativo.
- La red debe causar las mínimas molestias a los participantes que entren y salgan de ella.
- La red debe excluir a las unidades no C2 en las labores y responsabilidades de la Gestión de red.
- La red debe minimizar la Gestión dinámica.

El proceso de la Gestión de red se divide en 4 fases:

- Diseño, que incluye la especificación de los requisitos operativos y el diseño de la estructura de la red.
- Planeamiento previo, que es el conjunto de actividades a llevar a cabo previo a una operación. Incluye la selección del diseño de red a utilizar.
- Inicio, que es la inicialización de una plataforma y la entrada en la red.
- Operación, que incluye el control del funcionamiento de la red.

### 1.7.1 Diseño de redes.

El Diseño de redes Link-16 es el proceso mediante el cual se especifican los requisitos de comunicaciones en apoyo a operaciones tácticas determinadas, al tiempo que se traducen esos requisitos en el conjunto de datos de inicialización del terminal para su utilización por los participantes en la red.

Los terminales de Link-16 pueden integrarse en diferentes tipos de plataformas, cada una con distintos objetivos y misiones. Para posibilitar esta disparidad, los terminales incorporan en el software un número de parámetros intercambiables, que adaptan el terminal a la configuración de esa plataforma y su misión en cualquier operación. Todos estos parámetros del software se pueden cambiar, pero normalmente requiere una recompilación del software, lo que no es posible efectuarlo de forma inmediata.

Sin embargo, un subconjunto de esos parámetros puede ser cambiado cada vez que el terminal participa en una operación sin que se requiera una recompilación del software. Este subconjunto recibe el nombre de parámetros de inicialización. Los parámetros de inicialización se dividen en dos categorías:

a.- Aquellos que definen la participación de la plataforma en una red particular. El ejemplo más obvio es la asignación de *time slots* que utilizará la plataforma. Estos parámetros reciben el nombre de Parámetros Dependientes de la Red.

b.- Aquellos que no necesitan cambiarse para participar en una nueva red, denominados Parámetros Independientes de la Red. Un pequeño número de estos parámetros se establecen justo antes del comienzo de la operación. Por ejemplo, el Call Sign y el Número de Traza no se asignan hasta el proceso de Planeamiento Previo.

Una inicialización típica podría contener 550 parámetros, con un porcentaje 60/40 % entre los parámetros relativos a los *time slots* y los no relativos a los *time slots*. Por ejemplo, la definición de un bloque de *time slots* requiere unos 15 parámetros

individuales, y cada terminal puede mantener hasta un máximo de 64 bloques asignados.

El gran número de parámetros que necesita cada plataforma conduce a la necesidad de contar con un sistema automático que produzca los datos de inicialización. La producción centralizada de los diseños trata de mantener lo más sencillo posible los procedimientos de inicialización en las plataformas, ya que sus operadores solo operan con el Link-16, pero no necesitan saber en detalle los aspectos técnicos, como es la organización y asignación de *time slots*, por ejemplo.

Debido a que el proceso es complicado, los diseños se hacen con bastante anticipación a la operación a la que se pretende atender. No es un proceso que pueda hacerse justo antes de una operación. El primer paso dentro del proceso de diseño es analizar el entorno operacional y llegar a establecer el conjunto de requisitos de comunicaciones. Los requisitos se establecen en términos generales e incluyen lo siguiente:

- Entorno operacional global (paz/ejercicio/guerra).
- Número de plataformas (con previsión de participantes adicionales).
- Despliegue de las plataformas.
- Los datos a transmitir y recibir (con inclusión de la voz) por cada plataforma (o grupo de plataformas si así se facilita el diseño) y sus características:
    - Tipo (por función). Por ejemplo, vigilancia.
    - Cantidad.
    - Prioridad.
    - Seguridad, incluyendo requisitos de privacidad.
    - Resistencia a la perturbación.

Normalmente los requisitos deben establecerlos un grupo compuesto de operadores que definan los requisitos de comunicaciones en términos operativos apoyados por un experto capaz de traducir los requisitos operativos en decisiones de diseño de red. El experto tiene que ser capaz de ofrecer consejo al grupo, y dar respuesta inmediata sobre la viabilidad de los requisitos operativos específicos. Este grupo se

encarga de la coordinación de la producción, validación y distribución de los diseños de redes.

Los requisitos operativos tienen que ser analizados e interpretados en términos de requisitos del terminal. El primer paso es considerar el entorno operacional global, ya que las características de la transmisión del terminal pueden tener que restringirse para adaptarse al acuerdo de uso de frecuencias en vigor en el área de operaciones.

La cantidad y tipo de datos a transmitir se establece por cada NPG en el que participe cada plataforma, y se deriva de los requisitos de comunicaciones. Debe seleccionarse el método de acceso (Dedicado, por Contienda, etc.) que permita el adecuado flujo de datos. También hay que establecer los requisitos de seguridad ó privacidad para ajustar la asignación de las variables criptográficas, la utilización de redes apiladas y/o múltiples redes y la asignación de *time slots* específicos para restringir el acceso a ciertos datos. La utilización de múltiples redes puede ser apropiado para incrementar la capacidad de las funciones en las que no sea necesario que participen todas las plataformas.

Debe de examinarse de antemano el despliegue geográfico de los participantes para determinar los requisitos de relé. Esto incluye el análisis de la información que necesitará relé, de la plataforma que actuará de relé y si es aplicable el modo de operación relé a utilizar. Lo normal es que el examen esté basado en la geometría del horizonte radio y la distancia, pero puede tenerse también en consideración el perfil del terreno en la zona de operaciones y las posibles interferencias y perturbaciones.

A este nivel, los requisitos cripto se identifican de forma general, sin fijar de forma específica las claves. Esto se hace así porque el diseño se efectúa a largo plazo, por lo que no se puede identificar las claves exactas a utilizar en un día determinado. No obstante, en el diseño de red se identifican las claves a utilizar asignándoles una etiqueta lógica denominada CVLL. De esta forma se puede efectuar la división del tráfico de mensajes y asignar a algunas plataformas claves diferentes. La asociación de las CVLL con las claves reales específicas se efectúa en la etapa de planeamiento.

Los requisitos de comunicaciones resultantes se remiten al Centro de Diseño de Redes, que es el Centro nacional encargado de realizar los distintos diseños de redes Link-16. Para ello se cuenta con una herramienta software adecuada, que transforma los requisitos de comunicaciones en la asignación de *time slots* específicos para cada participante. En España se ha desarrollado una herramienta nacional, llamada ANITA (*Application for Network Implementation. Time slots Assignment*) en el Laboratorio Data Link del Departamento de Electrónica del CIDA (Centro de Investigación y Desarrollo de la Armada) [Vera05].

La asignación de *time slots* se valida de acuerdo con las reglas de validación que el terminal debe de aplicar. Después de esta validación, la herramienta de diseño de redes genera los ficheros de salida de datos en el formato apropiado para que pueda ser inicializada cada plataforma de la red. Estos ficheros de datos de inicialización contienen los parámetros que corresponden a la asignación de *time slots* más otros parámetros de cada participante. Cada conjunto de datos se adapta a los requisitos de comunicación de cada plataforma.

El diseño producido debe pasar ciertas comprobaciones:

- Comprobación para asegurar la consistencia de cada conjunto de datos de inicialización producido.
- Comprobación de validación del terminal, para asegurar que los terminales aceptarán los datos de inicialización.
- Comprobación de las restricciones para tiempo de paz para asegurar su conformidad.

Con objeto de facilitar la selección y modificación de los diseños de redes durante el proceso posterior de planeamiento previo, la herramienta de diseño de redes edita un "Resumen Descriptivo de la Red", el cual es una descripción de las características técnicas y operativas de la red. Este Resumen contiene como mínimo lo siguiente:

a. Nombre de la red. Las redes Link-16 tienen un convenio de designación mediante el cual las redes se identifican por 9 caracteres alfanuméricos.

b. Resumen. Se dará la suficiente información para describir completamente las características técnicas y operativas de la red.
c. Matriz de Conectividad. Se asignan los bloques de *time slots* a cada participante en la red para cada NPG en el que participe, según sus requisitos operativos.
d. Time-Line. Muestra la proporción de *time slots* asignados a cada NPG dentro de un *frame* (12 segundos, es la unidad de diseño de una red Link-16).
e. Mapa de la carga cripto. Este mapa asigna las CVLL a cada participante en la red.
f. *Time Slot Duty Factor* (TSDF). Muestra el cálculo del número de pulsos transmitido por cada plataforma y para el conjunto de la red.

Como es lógico, tiene que haber diseños para operaciones de paz, para ejercicios y para combate y, aunque los diseños se ajustan a escenarios específicos, pueden servir para otros escenarios más limitados. Por lo tanto, lo usual es contar con una librería o colección de diseños de redes. Obviamente, aunque puede haber un número infinito de variantes por cada escenario operacional, eso no significa que haya que producir un diseño por cada variante.

Solo deben de mantenerse en vigor unos cuantos diseños (10 ó menos). Como los diseños puede que no se adapten perfectamente al escenario operacional, tiene que haber cierta habilidad para seleccionar la librería de diseños que mejor cumpla la mayoría de requisitos de los escenarios previstos. Obviamente, existen ventajas en mantener limitado el número de diseños. En primer lugar se consigue sencillez en la producción logística, en la distribución y en el control de la librería. En segundo lugar, tanto los operadores como los operativos llegan a confiar y conocer los diseños que ya han utilizado. Esto es particularmente aplicable al Gestor de red.

La librería de redes debe contener diseños validados y aprobados para uso operativo. Esta aceptación debe incluir una prueba operativa limitada, seguida de una realimentación de los operadores. Así se tendrá una librería de diseños que funcionan correctamente mediante un proceso iterativo de mejora continua del diseño.

Para una correcta operación de la red es esencial el uso coordinado de las criptovariables. Por lo tanto, la disponibilidad de las claves es tan importante como la disponibilidad del diseño. Así, de la misma forma que es necesario que el diseño se

produzca con suficiente tiempo de antelación, las claves cripto deben de estar disponibles para cuando se decide efectuar la operación.

La gestión de claves de los terminales es ligeramente más complicada que la de otros equipos, ya que el equipo de Link-16 puede tener en operación hasta 4 claves al mismo tiempo. No obstante, es el diseñador el que decide cuantas claves se utilizarán para segregar la información dentro de la red. En el caso de que se autorizase a todos los participantes a compartir toda la información, sólo se utilizaría una clave.

### 1.7.2 Planeamiento previo de la misión.

El segundo proceso de la Gestión de redes es el Planeamiento de la Operación por un periodo determinado y para una operación concreta. El planeamiento previo incluye la selección del diseño a utilizar, la emisión del mensaje operativo OPTASK LINK, la designación de las claves criptográficas y otros detalles de planeamiento que permitan la participación en la red de todas las unidades de manera coordinada.

El planeamiento es normalmente responsabilidad de un Estado Mayor Operativo. No obstante, se puede delegar esta responsabilidad en un mando subordinado. Se debe tener en cuenta la disponibilidad de la unidad para llevar a cabo esta tarea, incluyendo el acceso a la información sobre los diseños de redes, el acceso a las criptovariables, y la disponibilidad de personal adiestrado.

La selección del diseño de red a utilizar debe basarse en el entorno operativo y en los requisitos de comunicaciones. Los criterios utilizados para la elección incluirán los siguientes:

- El diseño contiene como mínimo los tipos de cada plataforma requeridos para la operación.
- El diseño soporta el intercambio de información en términos de capacidad y NPGs planeados.
- El diseño contiene disponibilidad de plataformas relé (tiene la conectividad requerida).

- El diseño contiene la segregación de la información requerida.
- El diseño concuerda con el entorno previsto (paz, guerra, ejercicio) y respeta el acuerdo de uso de frecuencias.

La librería de diseños de redes contiene la suficiente información para permitir la rápida selección del diseño que mejor se adapte a los requisitos operativos específicos. El personal de planeamiento puede prever el uso de más de una red. Para facilitar el cambio entre una y otra, la plataforma puede cargar en el terminal más de un diseño. Un ejemplo típico de esta estructura es el caso en que se carga un diseño de adiestramiento en tiempo de paz y otro diseño para un ejercicio determinado.

El cambio de una red a otra no tiene por que implicar una reinicialización del terminal. Solo se necesitaría cambiar un conjunto de parámetros, al tiempo que se permanece sincronizado con el resto de participantes. Esta operación no debe de confundirse con el cambio a una red de otra estructura, que sí implicaría una salida de la red, una reinicialización del terminal y la sincronización a la nueva red.

En el planeamiento hay que asignar las funciones de las unidades participantes, como por ejemplo el *Network Time Reference* (NTR), el *Navigation Controller* (NC), el *Position Reference* (PR) y el *Initial Entry Joint Unit* (IEJU). Cualquier terminal puede desempeñar estas funciones. No obstante, es necesario seleccionar a las unidades antes de la operación. Además de estas funciones dentro de la red, hay que seleccionar también el sistema de navegación para las plataformas.

En la asignación de las funciones de las plataformas, se tendrán en cuenta los siguientes factores:

- Los sistemas de navegación de las unidades.
- La duración de la unidad en la zona de operaciones.
- La posición de la plataforma dentro del despliegue.

Después de seleccionar el diseño de la red, se tiene que evaluar la cantidad de criptovariables necesarias y escoger las apropiadas de la lista de claves disponibles. Esto

significa que se debe tener en conocimiento la distribución de criptovariables, ya que no sería viable seleccionar una clave que no esté disponible.

Por otra parte, el diseño contempla las criptovariables necesarias en la red etiquetadas como CVLL. Sin embargo, las claves reales tienen su propio nombre de referencia. Es obvio que se debe emitir una tabla que identifique cada etiqueta CVLL con una clave. Esta tabla se suele emitir en el mensaje OPTASK LINK.

El mensaje OPTASK LINK es el medio a través del cual el personal encargado del planeamiento pone en conocimiento de todos los participantes las decisiones tomadas. El OPTASK LINK ha sido modificado para incluir los datos necesarios de Link-16, y contiene datos como:

- Grupo fecha-hora en que se establecerá la red y periodo de funcionamiento.
- El nombre de la red a utilizar.
- La asignación de funciones (NTR, PR, NC y IEJU).
- La tabla de identificación CVLL - nombre de clave.
- La asignación de bloques de números de traza.
- Otras instrucciones de operación, como la utilización de filtros.

### 1.7.3 Iniciación de la red.

En esta fase, los datos genéricos de plataformas se traducen en datos de inicialización, y se preparan para la carga en el terminal de una plataforma concreta. Cada plataforma tiene su propio equipo específico para la carga de datos de inicialización en el terminal, a través de un soporte de carga (cartucho, tarjeta de memoria, etc.). Este sistema de carga recibe distinta denominación dependiendo de la plataforma a la que apoya, por lo que para simplificar la referencia se utiliza el término genérico de Dispositivo de Preparación de Datos de Inicialización (IDPF).

El IDPF tiene que aceptar el formato de ficheros recibido del Centro de Diseño de Redes y obtener los datos necesarios en el soporte apropiado para la carga del

terminal correspondiente. Además, el IDPF permite al operador adaptar los parámetros de inicialización a la misión específica.

Es usual que los Parámetros Independientes de la Red para el tipo de plataforma estén permanentemente cargados en el IDPF. No obstante, esto acarrea cierta responsabilidad en la gestión de la configuración para el mando de la unidad, ya que la misma plataforma puede operar desde varias localizaciones y, por lo tanto, los Parámetros Independientes de la Red tienen que ser consistentes. La operación del IDPF también envuelve la inserción de ciertos parámetros de inicialización, los cuales afectan a la operación de ese día.

Después de la operación del IDPF, se deben preparar los datos de inicialización para la misión específica, para lo cual hay que introducir ciertos parámetros concretos de la misión, como puede ser el *Call Sign*.

Resumiendo, el IDPF combina los siguientes datos para producir los datos de inicialización de la misión:

- Los Datos Independientes de la Red para el tipo de plataforma.
- Los Datos Dependientes de la Red para la red a utilizar.
- Cualquier parámetro introducido por el operador.

El fichero de datos que se obtiene se carga en el procesador de Data Link (equipo que procesa todas las señales de información de la plataforma y las convierte a Link-16 para ser enviadas por el terminal), y éste lo carga en el terminal. El operador facilita la carga a través de una secuencia de órdenes y mensajes de estado que aparecen en la consola del equipo. En este punto se pueden introducir algunos últimos parámetros.

Si la carga y los datos son correctos, el siguiente paso depende de la urgencia de la entrada en la red del terminal. Si se selecciona el parámetro *Start Net Entry*, el terminal intentará sincronizarse a la red. Si la entrada en la red no es inminente, se puede seleccionar el modo *Standby*, de forma que los datos cargados se pueden mantener por un periodo aproximado de 48 horas.

En el caso de que la carga o los datos no sean correctos, se pueden adoptar dos opciones. En primer lugar, intentar efectuar el proceso de carga de nuevo y, en segundo lugar, permitir que el terminal tome los valores por defecto de los parámetros no válidos causantes de la carga incorrecta.

Queda claro que los parámetros de inicialización que afectan a las variables criptográficas deben manejarse con sumo cuidado, ya que para participar en la red, después de efectuar la carga, el terminal debe utilizar la clave correcta para ese día. El proceso se complica más si la unidad tiene que participar en más de una red, con lo que deben cargarse en el terminal las claves para las redes en que se participa.

La carga de claves en el terminal se efectúa por medio de un equipo especial que transforma en primer lugar un segmento de papel a formato electrónico y, finalmente, su salida se utiliza para la carga de las claves en el terminal.

### 1.7.4 Operación de la red.

La filosofía que se persigue con la Gestión de red es que su funcionamiento debe de basarse en un buen diseño, en un buen planeamiento previo y en evitar la Gestión *on-line*. Sin embargo, el sistema Link-16 permite la reasignación de bloques de *time slots*, controlar la función de relé, controlar el uso de claves, e incluso facilitar claves *on-line*. También tiene previsto la corrección dinámica de algunos parámetros operativos. Pero esto no significa que todas las naciones implementen todas las opciones. Cada nación debe de decidir el nivel de sofisticación que quiere implementar en la gestión dinámica de red.

Otra problemática que se plantea es la elección de la plataforma que mejor efectúa las funciones de Gestión de red. Como es fácil de suponer, no existe la plataforma ideal. Por un lado, las aeronaves C2 suelen tener buena conexión con el resto de participantes, y por lo tanto pueden supervisar adecuadamente la red. Por el contrario, una aeronave no puede permanecer mucho tiempo en vuelo y, además, no siempre puede disponer de un operador en exclusiva para desempeñar la función de Gestión de red.

Una unidad terrestre o un barco tienen asegurada la permanencia en la red y, además, no suelen tener problemas a la hora de contar con un operador para la función de Gestión de red. Sin embargo, tienen difícil la supervisión de la red entera, debido a su limitación inherente de alcance visual.

Dentro del trabajo que le corresponde al Gestor de red se encuentra el supervisar el estado de la transferencia de información para vigilar los desequilibrios de asignación de *time slots* de algún participante y vigilar la cobertura de las unidades relé. En el caso de ser necesario, debe rectificar la situación transmitiendo los mensajes de Gestión de Red apropiados.

La Gestión dinámica de red debería de contener las siguientes capacidades:

- Supervisión de la conectividad de la red, del estado de las unidades y de la asignación de *time slots*.
- Control del acatamiento de las restricciones en vigor acordadas con las autoridades civiles.
- Mantenimiento de la calidad de la comunicación.
- Aseguración de la presencia continua de las funciones de red (NTR, etc.).
- Optimización de la conectividad a través de los relés.
- Ajuste de la capacidad de transmisión mediante la reasignación de *time slots*.
- Provisión *on-line* de las claves criptográficas.
- Cambio de uso de variables.
- Cambio de las estructura de la red.
- Organización del final de la operación.

## 1.8 Objetivos y organización de la tesis.

Como se ha comentado anteriormente en este capítulo, a cada participante en una red Link-16 se le asigna un grupo de intervalos de tiempo (*time slots*) para transmitir un determinado tipo de mensajes (grupo de participación). Pues bien, el objetivo de esta tesis es la mejora del algoritmo TSR (*Time Slot Reallocation*) de

reasignación dinámica de *slots* actual en dos situaciones posibles: que la suma de las demandas de bloques de *slots* de todos los terminales sea mayor que el número de bloques de *slots* disponible, y que la suma de las demandas de bloques de *slots* de todos los terminales sea menor que el número de bloques de *slots* disponible.

Para el modo de acceso TSR, en el caso de que la suma de las demandas de bloques de *slots* de todos los terminales sea mayor que el número de bloques de *slots* disponible, se propone la utilización de diversas técnicas de Optimización Combinatoria para la mejora del algoritmo TSR actual. Estas técnicas [Grotschel93, Papadimitriou82] han sido tratadas previamente en la literatura [Hopfield85, Kurokawa94, Movahhedinia95, Lazaro00, IEEE00, Chakraborty01] para aplicaciones civiles similares. Los métodos más comunes están basados en optimización y aprendizaje. Con el uso de estas técnicas, se pretende eliminar los casos de conflicto en la asignación y minimizar el valor medio de la cola máxima de mensajes no transmitidos en cada terminal participante, es decir, que para un número de *reallocation periods* dado, las colas de mensajes no transmitidos en todos los participantes en una red se mantengan lo mas pequeñas posibles.

En el caso de que la demanda total de *slots* sea menor que la capacidad disponible, a cada terminal se le asigna un número de bloques de *slots* igual al demandado. Sin embargo, quedan bloques de *slots* libres, sin asignar. Pues bien, en esta tesis se propone la utilización de Técnicas Predictivas y Teoría de Juegos para predecir esta demanda de *slots* en función de las necesidades en los últimos *reallocation periods*, asignando a cada terminal parte de los bloques de *slots* que quedan libres en la red, además de los demandados por él. Con ello se pretende minimizar el tiempo de transmisión de los mensajes de cada terminal desde que se demandan los *slots* en un *reallocation period* hasta que se transmiten, ya que, al asignarle más bloques de *slots*, puede transmitir antes en el siguiente *reallocation period*.

Esta tesis doctoral se estructura en cuatro partes y un apéndice:

- La Parte I consiste en la introducción y revisión de los conceptos que se van a manejar en esta tesis. Consta de cuatro capítulos:

- El Capítulo 1 es una introducción general a las redes de comunicaciones Link-16, y presenta la motivación de la tesis y una descripción de su contenido.
- El Capítulo 2 presenta los conceptos fundamentales y problemática de las Redes Neuronales y, más concretamente, de las Redes de Hopfield.
- El Capítulo 3 presenta los conceptos fundamentales en lo que se refiere a Algoritmos Genéticos.
- El Capítulo 4 presenta los conceptos fundamentales en lo que se refiere a Técnicas Predictivas y Teoría de Juegos.

- La Parte II es el núcleo de esta tesis, y contiene las propuestas de mejora del algoritmo TSR de reasignación dinámica de *slots*. Consta de dos capítulos:
  - El Capítulo 5 presenta diversas propuestas basadas en Técnicas de Optimización Combinatoria para el caso en que la demanda total de *slots* exceda de la capacidad disponible.
  - El Capítulo 6 presenta las propuestas para el caso en que la demanda total de *slots* sea menor que la capacidad disponible. Estas propuestas están basadas en Técnicas Predictivas y Teoría de Juegos.

- La Parte III corresponde a la interpretación y discusión de los resultados obtenidos en las simulaciones realizadas para las propuestas descritas en la parte anterior. Consta de dos capítulos:
  - El Capítulo 7 presenta un análisis y discusión de los resultados obtenidos en las simulaciones realizadas para el caso en que la demanda total de *slots* exceda de la capacidad disponible, así como una comparación entre las diversas propuestas.
  - El Capítulo 8 presenta un análisis y discusión de los resultados obtenidos en las simulaciones realizadas para el caso en que la demanda total de *slots* sea menor que la capacidad disponible, así como una comparación entre las diversas propuestas.

- La Parte IV, que consta únicamente del Capítulo 9, corresponde a la presentación de las conclusiones globales de esta tesis y la propuesta de líneas de investigación futuras para realizar a partir de dichas conclusiones.

- El Apéndice A describe con exactitud el algoritmo actual de selección y asignación de bloques de *slots* a cada terminal participante en una red Link-16 en el modo de acceso TSR de reasignación dinámica de *slots*.

# Capítulo 2 Redes Neuronales

## 2.1 Introducción.

La asignación dinámica de intervalos de tiempo en redes de comunicaciones tácticas militares es un problema de optimización combinatoria. Para este tipo de problemas conviene (con mayor o menor intensidad) disponer de soluciones válidas en un tiempo corto. Por otra parte, el carácter de algunos problemas combinatorios implica alta probabilidad de obtener (aleatoriamente) una solución cercana al óptimo absoluto [Rhee91]. Esta característica es la principal motivación del uso de determinadas redes neuronales, que pueden aportar soluciones más rápidas que un ordenador con un algoritmo eficiente, y suficientemente buenas para un problema de este tipo.

El término red neuronal artificial engloba un gran número de esquemas de tratamiento de información que mantienen cierta inspiración en su contrapartida natural, suponiendo una alternativa al uso de ordenadores convencionales en numerosas aplicaciones [Alspector93]. La interacción de múltiples procesadores simples en paralelo, llamados neuronas, constituye el principio de su funcionamiento [Hertz91]. Su potencia de cálculo es debida, primero, a su estructura en paralelo masivamente distribuida y, segundo, a su capacidad de aprendizaje y generalización. Esta generalización se refiere a la capacidad de la red neuronal de producir salidas razonables para entradas no especificadas durante el entrenamiento (aprendizaje). Estas dos capacidades les hacen posible resolver problemas complejos que normalmente son intratables. Estos problemas se descomponen en un número de tareas relativamente simples, y se asignan redes neuronales a cada subconjunto de tareas que mejor se

adaptan a las capacidades inherentes de las redes [Haykin99]. Por otra parte, es importante reconocer que queda un largo camino por recorrer antes de que se pueda construir una arquitectura cuyo comportamiento sea igual al del cerebro humano.

Como características más destacables de las redes neuronales han de señalarse las siguientes:

- No linealidad. Su funcionamiento es intrínsecamente no lineal, lo que da lugar a dinámicas variadas y, a la vez, dificulta su comprensión.

- Entrenamiento. Tienen la capacidad de aprender del entorno presentándoles un conjunto de entradas con sus correspondientes salidas deseadas. Los pesos de las redes se modifican para minimizar la diferencia entre la salida deseada y la actual. El entrenamiento se repite hasta que la red alcanza un estado en el que los pesos no cambian. Así, la red aprende de los ejemplos construyendo un mapa de entrada-salida para el problema.

- Adaptatividad. Tienen capacidad de adaptarse al entorno o aprender de él.

- Generalización. Muchas pueden aproximar cualquier función, lo que teóricamente les da capacidad de clasificación genérica.

- Robustez. La redundancia en la arquitectura y su especial representación de la información proporcionan robustez al esquema.

- Información contextual. El conocimiento viene representado por la estructura y el estado de activación de la red. Cada neurona de la red está potencialmente afectada por la actividad global del resto de neuronas en la red. En consecuencia, la información contextual es tratada con naturalidad por la red neuronal.

- Tolerancia a fallos. Una red neuronal, implementada de forma *hardware*, tiene el potencial de ser inherentemente tolerante a fallos, o la capacidad de cálculo robusto,

en el sentido de que su funcionamiento se degrada fácilmente bajo condiciones operativas adversas antes que producir un fallo catastrófico.

- Implementación VLSI. Son fácilmente implementables en *hardware* (VLSI), lo que permite, junto con el paralelismo masivo, disponer de dispositivos muy rápidos para determinadas tareas.

## 2.2 Modelos de neurona.

Una neurona es una unidad de procesado de información fundamental en el funcionamiento de una red neuronal. El diagrama de bloques de la Fig.2.1 muestra el modelo de una neurona, que representa la base para el diseño de redes neuronales. En este modelo neuronal se identifican tres elementos básicos [Haykin99]:

- Un conjunto de enlaces, cada uno de los cuales se caracteriza por un peso $w$. Específicamente, una señal $x_j$ a la entrada de un enlace j conectado a la neurona k es multiplicada por el peso $w_{kj}$, que puede tener valores positivos o negativos.

- Un sumador para sumar las señales de entrada multiplicadas por sus respectivos pesos. Estas operaciones constituyen un combinador lineal.

- Una función de activación para limitar el rango de amplitud de la salida de una neurona a algún valor finito. Típicamente, el rango de amplitud normalizado de la salida de una neurona es el intervalo cerrado [0,1] o [-1,1].

Este modelo neuronal también incluye un sesgo o desviación aplicado externamente, llamado $b_k$, cuya función es incrementar o disminuir la entrada de la función de activación, dependiendo de que sea positiva o negativa.

De forma matemática, se puede describir una neurona k mediante las siguientes ecuaciones:

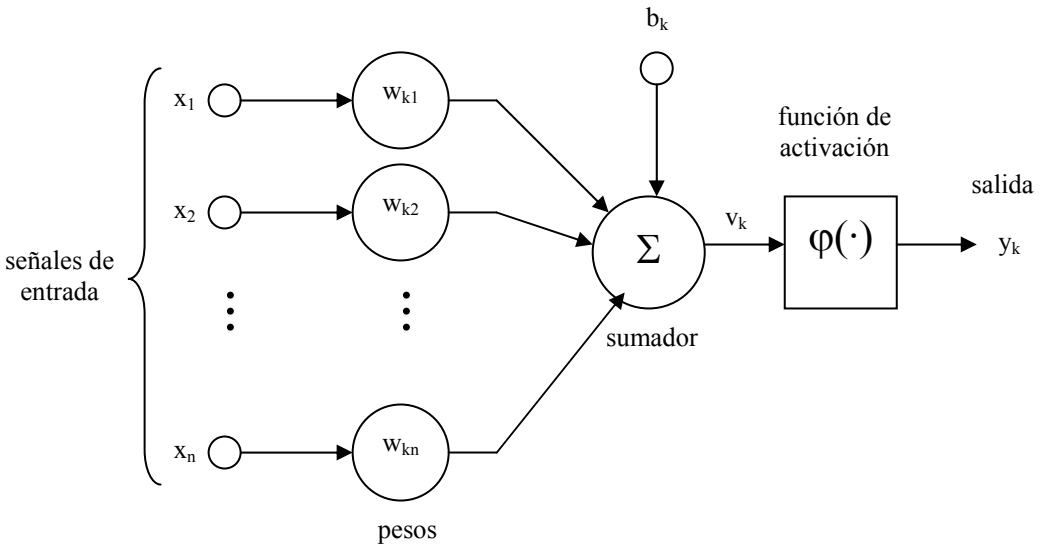



Fig.2.1: Modelo no lineal de una neurona

$$u_k = \sum_{j=1}^{n} w_{kj} x_j \tag{2.1}$$

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

donde $x_1$, $x_2$, ..., $x_n$ son las señales de entrada; $w_{k1}$, $w_{k2}$, ..., $w_{kn}$ son los pesos de la neurona; $u_k$ es la salida combinación lineal de las señales de entrada; $b_k$ es el sesgo; $\varphi(\cdot)$ es la función de activación; y $y_k$ es la señal de salida de la neurona. El sesgo $b_k$ sirve para aplicar una transformación afín a la salida $u_k$ del combinador lineal del modelo neuronal del siguiente modo:

$$v_k = u_k + b_k \tag{2.3}$$

Dependiendo de que $b_k$ sea positivo o negativo, la relación entre el campo local inducido o potencial de activación $v_k$ de la neurona k y la salida $u_k$ del combinador lineal es la mostrada en la Fig.2.2.

La combinación de las ecuaciones (2.1) a (2.3) se puede formular de la siguiente forma:

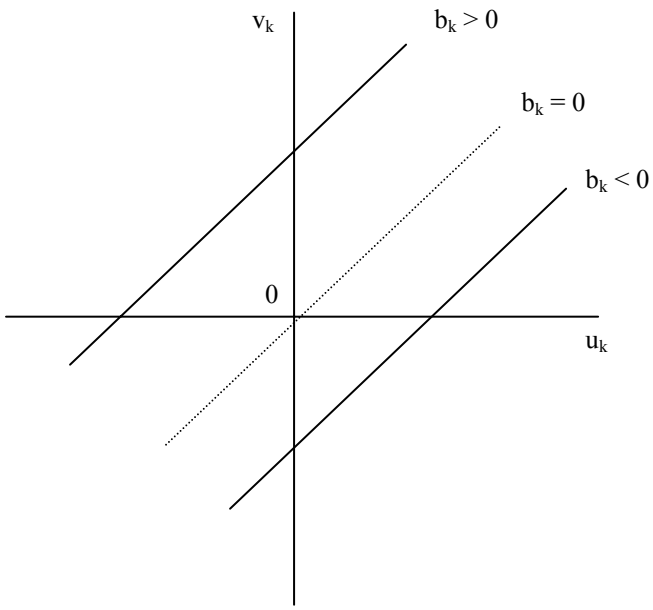



Fig.2.2: Transformación afín producida por la presencia de un sesgo

$$v_k = \sum_{j=0}^{n} w_{kj} x_j \tag{2.4}$$

$$y_k = \varphi(v_k) \tag{2.5}$$

En la ecuación (2.4) se ha añadido un nuevo enlace, cuya entrada es

$$x_0 = +1 \tag{2.6}$$

y su peso es:

$$w_{k0} = b_k \tag{2.7}$$

Con ello, se puede reformular el modelo neuronal añadiendo una nueva señal de entrada fija a +1, y añadiendo un nuevo peso igual al sesgo $b_k$.. Este modelo es matemáticamente equivalente al anterior.

### 2.2.1 Tipos de función de activación.

La función de activación $\varphi(v)$ define la salida de una neurona en relación con el campo local inducido $v$. Se identifican tres tipos básicos de función de activación:

- Función umbral. Para este tipo de función de activación se tiene que

$$\varphi(v) = \begin{cases} 1 & \text{si } v \geq 0 \\ 0 & \text{si } v < 0 \end{cases} \tag{2.8}$$

Correspondientemente, la salida de la neurona k que emplea esta función umbral se expresa como

$$y_k = \begin{cases} 1 & \text{si } v_k \geq 0 \\ 0 & \text{si } v_k < 0 \end{cases} \tag{2.9}$$

donde $v_k$ es el campo local inducido de la neurona, es decir,

$$v_k = \sum_{j=1}^{n} w_{kj} x_j + b_k \tag{2.10}$$

Este tipo de neurona es referido en la literatura como el modelo McCulloch-Pitts, en reconocimiento al trabajo realizado por McCulloch y Pitts [McCulloch43]. En este modelo, la salida de una neurona toma el valor 1 si el campo local inducido de esa neurona es no negativo, y 0 en el resto de casos.

- Función lineal por tramos. Para este tipo de función de activación se tiene que

$$\varphi(v) = \begin{cases} 1 & \text{si } v \geq +\frac{1}{2} \\ v & \text{si } +\frac{1}{2} > v > -\frac{1}{2} \\ 0 & \text{si } v \leq -\frac{1}{2} \end{cases} \tag{2.11}$$

donde se asume que el factor de amplificación en la región lineal de operación es la unidad. Se puede ver esta función como una aproximación a un amplificador no lineal. Si el factor de amplificación en la región lineal de operación se hace infinitamente grande, esta función se reduce a la función umbral.

- Función sigmoide. Esta tipo de función, cuya gráfica tiene forma de "s", es el más usado en la construcción de redes neuronales artificiales. Un ejemplo es la función definida por

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{2.12}$$

donde *a* es la pendiente de la función sigmoide. Variando este parámetro, se obtienen funciones de diferentes pendientes. En el límite, cuando este parámetro se aproxima a infinito, la función sigmoide se convierte en una función umbral. Mientras que la función umbral asume los valores de 0 o 1, una función sigmoide asume un rango continuo de valores entre 0 y 1. Hay que hacer notar también que la función sigmoide es diferenciable, mientras que la función umbral no, característica importante en la teoría de redes neuronales.

Las funciones de activación definidas en las ecuaciones (2.8), (2.11) y (2.12) están comprendidas en un rango entre 0 y +1. Algunas veces es deseable tener una función de activación en un rango entre –1 y +1, en cuyo caso la función tiene una forma asimétrica con respecto al origen; es decir, es una función impar del campo local inducido. Específicamente, la función umbral de la ecuación (2.8) es definida como

$$\varphi(v) = \begin{cases} 1 & \text{si } v > 0 \\ 0 & \text{si } v = 0 \\ -1 & \text{si } v < 0 \end{cases} \tag{2.13}$$

que es llamada la función signo. Para la forma correspondiente de una función sigmoide se puede usar la función tangente hiperbólica, definida por

$$\varphi(v) = \tanh(v) \tag{2.14}$$

### 2.2.2 Modelo estocástico de neurona.

El modelo neuronal descrito en la Fig.2.1 es determinístico en el sentido en que su comportamiento entrada-salida es definido de forma precisa para todas las entradas. Para algunas aplicaciones, es recomendable basar el análisis sobre un modelo neuronal estocástico. En una primera aproximación, a la función de activación del modelo

McCulloch-Pitts se le puede dar una interpretación probabilística. Específicamente, una neurona puede estar en dos estados posibles: +1 o –1. La decisión para una neurona de conmutar de un estado al otro es probabilística. Sea *x* el estado de la neurona, y *P(v)* la probabilidad de conmutar, donde *v* es el campo local inducido de la neurona. Entonces, se puede escribir

$$x = \begin{cases} +1 & \text{con probabilidad } P(v) \\ -1 & \text{con probabilidad } 1 - P(v) \end{cases}$$

Para la función *P(v)* se puede elegir la función sigmoide

$$P(v) = \frac{1}{1 + \exp(-v/T)} \tag{2.15}$$

donde *T* es una pseudotemperatura utilizada para controlar el nivel de ruido y, por consiguiente, la incertidumbre en la conmutación de la neurona. Por otra parte, hay que hacer notar que *T* no es la temperatura física de una red neuronal, sea biológica o artificial. Más bien, se debería interpretar como un parámetro que controla las fluctuaciones térmicas que representan los efectos de un ruido. Es más, cuando *T* tiende a cero, el modelo estocástico de neurona descrito por la ecuación (2.15) se reduce a un modelo sin ruido (es decir, determinista), que es el modelo McCulloch-Pitts.

## 2.3 Redes neuronales en optimización combinatoria.

Para atacar este tipo de problemas, las aproximaciones a los problemas de optimización combinatoria, en relación con las redes neuronales, son variadas [Cichocki93]. Se presenta en este apartado una recopilación de los métodos y las técnicas más comunes:

- Red de Hopfield analógica: su característica fundamental es la fuerte realimentación entre neuronas; se pretende que la dinámica del sistema resultante converja a las soluciones buscadas [Hopfield82]. Esta red se describirá con detalle en la Sección 2.4 de este capítulo.

- "*Mean Field Annealing*" (MFA): este esquema surge de una aproximación determinista al equilibrio de un sistema estocástico de elementos magnéticos en interacción. Este método, a una temperatura dada, da lugar a unas ecuaciones análogas a las de la red de Hopfield analógica [Hertz91]. Se han propuesto diferentes algoritmos de gestión de la temperatura [Aarts89] que permiten obtener mejores soluciones, en detrimento del tiempo necesario para alcanzar la convergencia.

- Redes competitivas: se describen por unas ecuaciones muy parecidas a las de la red de Hopfield analógica [LiT90]; la mayor diferencia estriba en la función entrada-salida de las neuronas.

- Enfriamiento simulado ("*Simulated Annealing*"): emulan procesos físicos de temple. Sus resultados son posiblemente los de mayor calidad, aunque requieren mucho tiempo de cálculo [Aarts89], por lo que son adecuados cuando se prima la calidad de la solución frente al tiempo de obtención de la misma.

- Métodos híbridos de los anteriores. Entre estos métodos, cabe destacar el de la red de Hopfield y el de enfriamiento simulado que da lugar a la máquina de Boltzmann [Aarts89, Hertz91]. La máquina de Boltzmann introduce neuronas ocultas (que no aparecen directamente en la salida del sistema) y permite un aprendizaje basado en enfriamiento simulado. Debido a sus mayores grados de libertad, parece obvio que la máquina de Boltzmann puede ofrecer ventajas en cuanto a optimización, pero a costa de un mayor tiempo de procesado, debido al mayor número de neuronas y al método de aprendizaje, inherentemente lento.

## 2.4 La red de Hopfield.

De la discusión previa sobre esquemas neuronales genéricos para optimización combinatoria, es obvio el compromiso entre la optimalidad de las soluciones, complejidad y velocidad. Por otra parte, también se desea rapidez en la obtención de soluciones. Esto nos hace proponer inicialmente la aplicación del esquema de Hopfield.

La característica fundamental de una red neuronal de tipo Hopfield es la fuerte realimentación entre sus neuronas [Hopfield85]; se pretende que la dinámica del sistema resultante converja a las soluciones buscadas. Una ventaja de este tipo de redes neuronales es la rapidez de sus implementaciones *hardware*, y sus ecuaciones son base de esquemas más elaborados, que se podrían desarrollar según las necesidades de la aplicación y el conocimiento de las limitaciones de esta red. No obstante, esta red neuronal también tiene algunos inconvenientes:

- La dificultad en determinar los coeficientes de la función de energía para obtener soluciones válidas.

- Las prestaciones en la optimización son, en general, bajas comparadas con algunos esquemas alternativos (clásicos o emergentes).

- Existe además un compromiso entre la optimización y la validez de las soluciones al fijar los parámetros de la red.

Es posible fijar restricciones a estos parámetros para conseguir soluciones válidas [Bousoño95]. La ventaja de esta aproximación es la sencillez, con respecto a otras aproximaciones [Aiyer90], de las ecuaciones resultantes que permiten la discusión de la influencia de los distintos parámetros en la validez y la velocidad del esquema.

### 2.4.1 Problemática de la red de Hopfield.

Una red de Hopfield consiste en un conjunto de neuronas y su correspondiente conjunto de retardos, formando un sistema realimentado por múltiples lazos, como se muestra en la Fig.2.3. El número de lazos de retardo es igual al número de neuronas. Básicamente, la salida de cada neurona es realimentada, pasando por el elemento de retardo, al resto de neuronas en la red [Haykin99]. En otras palabras, sólo se excluye la auto-realimentación.

Las neuronas son procesadores sencillos caracterizados por una entrada $u$, una salida $v$ y una función de trasferencia $\phi$, tal que $v = \phi(u)$. La entrada de la neurona $i$, $u_i$,

se calcula linealmente a partir de las salidas de las neuronas conectadas a ellas mediante enlaces incidentes:

$$u_i = \sum_p w_{pi} v_p - \theta_i \qquad (2.16)$$

donde $w_{pi}$ es el peso del enlace que va desde la neurona $p$ hasta la neurona $i$, y $\theta_i$ es un umbral fijo aplicado externamente a la neurona $i$.

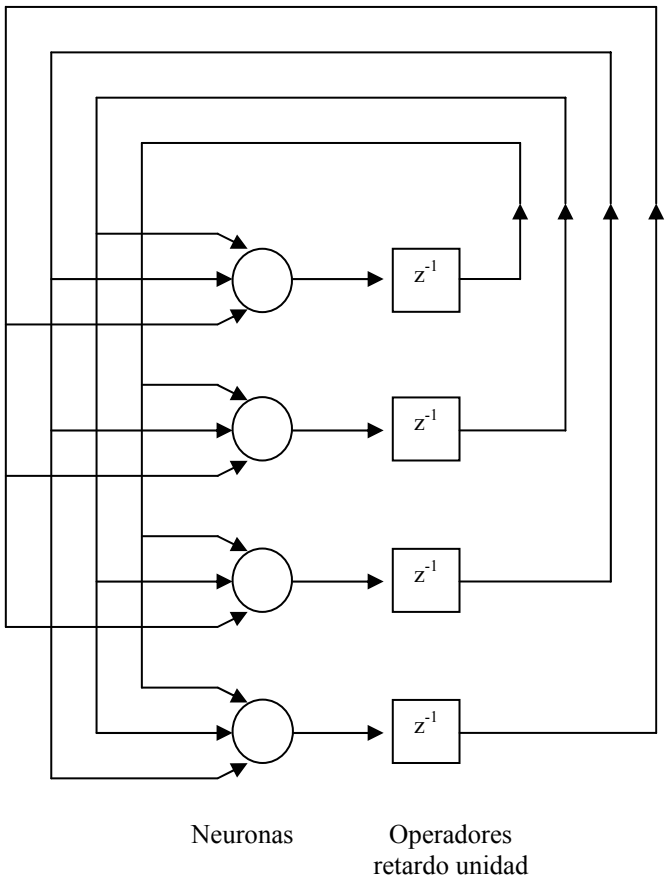



Fig.2.3: Arquitectura de una red de Hopfield de 4 neuronas

En el modelo básico, las neuronas son del tipo McCulloch-Pitts: la salida de la neurona $i$ es discreta, tomando valores en el conjunto {-1,1}, y la función de transferencia es de la forma

$$\phi(u_i) = \begin{cases} -1 & \text{si } u_i < 0 \\ 1 & \text{si } u_i \geq 0 \end{cases}$$

El funcionamiento de esta red se basa en el cambio de estado (o salida) de las neuronas que la forman. Estos cambios se producen en instantes discretos, $n$ = 1,2,3,..., y en cada instante se actualiza una neurona $i$ escogida aleatoriamente entre todas las neuronas:

$$v_i(n+1) = \phi(u_i(n)) \qquad (2.17)$$

donde la entrada $u_i(n)$ depende de los estados del resto de las neuronas del sistema en el instante $n$ según la ecuación (2.16). La evolución continúa hasta que se produce la convergencia: $v_i(n + 1) = v_i(n)$, $\forall i$, es decir, hasta que ninguna neurona del sistema se ve obligada a cambiar su valor de salida.

La aplicación típica de esta red es la de actuar como memoria asociativa. Una memoria asociativa es un sistema capaz de "recordar" una estructura completa a partir de la presentación de una de sus partes [Kohonen88]. En el caso de la red de Hopfield, las estructuras a "recordar" son vectores con componentes en {-1,1}.

Supongamos una memoria asociativa con dos vectores almacenados, v1 y v2. Supongamos igualmente que esa memoria es ideal, en el sentido de que sólo "recuerda" uno de los dos vectores, v1 ó v2. Si a este sistema se le presenta como entrada un vector con componentes aleatorias en {-1,1}, debería proporcionar como salida el vector v1 ó v2 que más cerca estuviera del vector de entrada, entendiendo por distancia entre dos vectores el número de componentes en que difieren (distancia de *Hamming*).

En la aplicación de la red de Hopfield como memoria asociativa, a cada componente del vector que se quiere representar se le hace corresponder una neurona. Los vectores a "recordar" se codifican en los pesos de la red. La presentación a la red de un vector aleatorio se realiza inicializando las neuronas con las componentes del vector que les correspondan. El funcionamiento esperado es la evolución de la red hasta la convergencia en el vector v1 ó v2 más cercano (en el sentido de *Hamming*) al presentado.

La problemática que surge en esta aplicación es genérica para el funcionamiento de esta red neuronal:

- El mecanismo de representación de los vectores a "recordar" reside en los pesos de la red, pero no es obvio cómo seleccionar éstos para almacenar unos vectores dados.

- El problema de la estabilidad y convergencia: cómo asegurar que, ante la inicialización con un vector cualquiera, la red evolucione hasta un vector dado, en lugar de permanecer oscilando o con otro tipo de dinámica no estable.

- La validez de las soluciones: que el vector "recordado" sea uno de los almacenados previamente y no una solución espúrea.

- La calidad de la memoria: que el vector "recordado" sea, de entre los almacenados, el más próximo al de inicialización.

La aplicación de la red de Hopfield a la optimización combinatoria implica, en primer lugar, que la red converja en aquellos vectores que sean solución válida del problema a resolver y, en segundo lugar, que seleccione entre éstos el de menor coste, o al menos, uno de coste pequeño respecto al de los demás.

Esta aplicación conlleva, por tanto, problemas análogos a los que surgen en la aplicación de memoria asociativa, cambiando el de la calidad de la memoria por el de la optimalidad de las soluciones obtenidas. La solución a todos estos problemas está en la determinación de los pesos de la red.

### 2.4.2 Dinámica de la red de Hopfield.

Una red de Hopfield puede verse como un sistema de ecuaciones diferenciales (si es continua) o en diferencias (si es discreta) no lineales, que pueden ser deterministas o estocásticas, según el carácter de las neuronas. La dinámica de la red está relacionada con el carácter de las soluciones de las ecuaciones que constituyen la red neuronal.

El estudio y caracterización de dichas soluciones dan lugar a una serie de conceptos y contribuciones que pueden clasificarse en estudios de dinámica

determinista [Aiyer90, LiJH88, Michel89] y estocástica [Hertz91, Sherrington93, Coolen93].

Las principales aportaciones del estudio de la dinámica estocástica son el diagrama de fases [Hertz91, Haykin99], que hace una partición del espacio de parámetros de la red estocástica en regiones con sistemas que proporcionan soluciones válidas y no válidas, y la capacidad de la red [Hertz91, McEliece87, Burshtein94], es decir, el número de patrones aleatorios que es capaz de almacenar.

No obstante, la mayor parte de la investigación realizada desde la primera aplicación explícita de esta red a problemas de optimización combinatoria [Hopfield85] se refiere a la red de Hopfield continua y determinista, acuñándose el término neurodinámica a tal efecto y abordando fundamentalmente el problema de la estabilidad del sistema [Haykin99]. El objetivo es calcular los parámetros, o imponer restricciones en los mismos, de tal forma que se solucionen los siguientes problemas:

- Problema de estabilidad: que las soluciones del sistema converjan a puntos concretos del espacio de fases, llamados atractores del sistema.

- Problema de la representación: que los atractores coincidan con los patrones a "recordar", para la aplicación de memorias asociativas, o con las soluciones válidas, para el problema de optimización.

- Problema de las soluciones espúreas: que no existan otros atractores distintos a los deseados y se eviten puntos que, sin tener el carácter de atractor, puedan atrapar el sistema (estancamiento).

Para asegurar la estabilidad, se puede imponer la restricción de simetría a los pesos de la red, es decir, $w_{pi} = w_{ip}$. Con esto, la evolución del sistema se restringe al decrecimiento monótono de la función de energía, o *Lyapunov* [Haykin99], cuya forma genérica es

$$E_{hop} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}V_iV_j - \sum_{i=1}^{n}\theta I_i \qquad (2.18)$$

donde *Vi* y *Vj* son las salidas de la neuronas *i* y *j* respectivamente, y los términos $\theta_i I_i$ representan un sesgo o desviación.

En cuanto a la selección de parámetros, generalmente se hace mediante prueba y error, aunque se complementa con estudios para determinar las regiones donde los parámetros proporcionan soluciones válidas, y su influencia en otros aspectos de la aplicación concreta que guíen esa selección.

La aproximación más extendida a la determinación de las restricciones para los parámetros de la red consiste en la linealización de las ecuaciones [Abe89, Aiyer90, Rong91]. La determinación de la relación entre los autovalores del sistema y los parámetros de la función de energía permite imponer restricciones para la validez de las soluciones. Esta aproximación plantea dos problemas: en primer lugar, se emplea una aproximación a las ecuaciones (linealización) de carácter local, cuya limitación para extraer conclusiones globales no está clara; en segundo lugar, la dificultad en el cálculo de los autovalores y la imposición de las restricciones sobre ellos dependen en gran medida del problema a resolver. Debido a esto último, es frecuente el uso de heurísticos adaptados al problema objeto de la aplicación [Funabiki92, Ali93].

Otra aproximación [Amin94] plantea las condiciones para la estabilidad de las soluciones válidas como atractores del sistema. Es, por tanto, un método global, dependiente de la descripción de las soluciones válidas en el hipercubo unidad. Las ecuaciones resultantes son sencillas de estudiar, lo que permite vislumbrar su influencia en aspectos tales como la velocidad de la red neuronal.

### 2.4.3 Representación de la información en la red de Hopfield.

El primer problema apuntado en la presentación de la red de Hopfield es la representación de los vectores a través de los pesos de la red $w_{ij}$. En este apartado se revisan las soluciones propuestas para la aplicación de optimización combinatoria.

Para este caso, el proceso de representación implica la codificación en los pesos de la red del conjunto de soluciones válidas del problema. Para ello, el proceso habitual consta de los siguientes pasos:

- La formulación del problema en variables 0-1, es decir, en variables $V_{ij} \in \{0,1\}$ que puedan describir el problema.

- La reducción de las restricciones sobre las variables 0-1 a términos cuadráticos (en estas variables) de penalización [Minoux86]. Este proceso consiste en formular una función cuadrática que se anule en aquellos valores de las variables que cumplan las restricciones, y sean distintos de cero y positivos en aquellos otros que no las cumplan. Generalmente, la forma de realizar esta reducción no es conocida, sino que es objeto de propuestas heurísticas que satisfacen estos objetivos con mayor o menor fortuna.

- La construcción de una función $E_p$ que sea la suma ponderada de los términos que representan las restricciones. Los coeficientes de la ponderación se denominarán parámetros.

- La identificación de $E_p$ con la función de *Lyapunov* de la red de Hopfield, definida en la ecuación (2.18), permite extraer el valor de los pesos de la red en función de los parámetros.

- La determinación del valor de los parámetros para proporcionar la estabilidad de las soluciones del problema y la convergencia a estas soluciones exclusivamente.

Existen otras propuestas para calcular los pesos de la red [Peterson89]. En general, se trata de un problema abierto en el que se usan heurísticos que dependen del problema que se trate, existiendo incluso formulaciones diferentes para la misma aplicación (por ejemplo, en el problema del camino más corto [Fritsh92]).

Pero en los problemas de optimización combinatoria, aparte de las restricciones impuestas al conjunto de los vectores, ha de proporcionarse a la red neuronal la

información sobre la función de coste. El método clásico de codificar esta información en la red neuronal consiste en utilizar la función de *Lyapunov* del sistema, completando la función $E_p$ descrita anteriormente con la suma de la función de coste que se pretende optimizar, y determinando los parámetros para conseguir cumplir las restricciones y la optimización de dicho coste.

Aún en el caso de que no hubiese restricciones sobre los vectores representados por la estructura neuronal, aparecen dos problemas:

- La codificación sólo es inmediata en el caso de problemas de minimización lineales o cuadráticos, con funciones de restricción lineales o cuadráticas (en el caso de maximización lineal o cuadrática, bastaría con considerar una transformación simple del coste).

- Los mínimos de la función de energía que encuentra la red son locales, trasladando la dificultad del problema de partida al de selección del estado inicial [Uesaka91].

La codificación conjunta de las restricciones y la función de coste en los parámetros de la función de *Lyapunov* supone una especie de "*aliasing*" entre ambos términos: no es posible determinar de forma objetiva qué parte del resultado de la suma corresponde a cada término (restricción u optimización) y, obviamente, la información sobre ambos aspectos es equívoca para el propio sistema. Esto de lugar a un problema típico de estas redes: el compromiso entre el cumplimiento de las restricciones y la optimalidad de las soluciones encontradas.

Si bien el compromiso entre la validez y la optimalidad de las soluciones es reconocido ampliamente en la literatura, lo es menos su relación con la velocidad y complejidad del sistema. Es evidente que, dada una función de *Lyapunov* con parámetros que favorezcan el cumplimiento de las restricciones, se puede compensar la falta de optimalidad en las soluciones con múltiples inicializaciones diferentes del sistema, dando lugar a un incremento en el tiempo de respuesta (si se realiza con el mismo sistema) o en la complejidad (si se consideran múltiples sistemas en paralelo). Otra versión de este compromiso se aprecia en los diferentes sistemas realizados para una misma aplicación, como en el problema del camino más corto, donde se consiguen

redes neuronales [Ali93] que cumplen las restricciones y consiguen resultados óptimos frente a otras redes [Bousoño1] que cumplen las restricciones, consiguen resultados de inferior calidad pero son más rápidas.

# Capítulo 3 Algoritmos Genéticos

## 3.1 Introducción.

Los Algoritmos Genéticos son algoritmos de búsqueda basados en los mecanismos de selección natural y genética natural. Combinan la supervivencia del mejor entre distintas estructuras de cadenas con el intercambio de información estructurada y aleatoria para formar un algoritmo de búsqueda similar al comportamiento humano. En cada generación se crea un nuevo conjunto de individuos (cadenas) usando fragmentos del mejor individuo de la generación anterior. Ocasionalmente se prueba con nuevos fragmentos para mejorar los individuos de la generación. Aunque los Algoritmos Genéticos no se basan solo en una búsqueda aleatoria. También explotan de forma eficiente la información histórica para especular sobre nuevos puntos de búsqueda que puedan mejorar el proceso.

El uso de estos algoritmos como métodos de optimización y exploración global comienza a mediados de los 70 con los trabajos de John Holland [Holland75]. Pero el que se considera texto básico en los Algoritmos Genéticos es "*Genetic Algorithms in Search, Optimization and Machine Learning*" [Goldberg89], escrito por David Goldberg en 1989. Es a partir de esta obra cuando se comienza a trabajar con más interés en esta área. Otras áreas de exploración en Algoritmos Genéticos corresponden a la programación genética [Michalewicz92, Koza92], que consiste en la optimización de programas usando los principios de la optimización global de los Algoritmos Genéticos, a través de la modificación de su propio código de programa.

## 3.2 Los Algoritmos Genéticos como métodos de búsqueda y optimización global.

El método de búsqueda global de los Algoritmos Genéticos se basa en la teoría Neo-Darwiniana de la evolución de las especies. Para los Algoritmos Genéticos, el espacio de búsqueda está compuesto por individuos con una cierta capacidad de resolver el problema planteado. De este espacio de búsqueda se extrae un número de individuos (supuestas soluciones, puntos del espacio de búsqueda) mucho menor que el existente en este espacio, con los que se construye la población inicial del entrenamiento. Basados en las teorías evolutivas, el individuo más apto, es decir, aquel que obtenga la mejor solución al problema planteado, tendrá mayor probabilidad de reproducirse y mejorar la población. Esta población, después de varias generaciones, debe estar compuesta por individuos que solucionen razonablemente bien el problema.

Los Algoritmos Genéticos difieren de los métodos de búsqueda tradicionales en varios aspectos [Goldberg89]:

- Los Algoritmos Genéticos no trabajan con el conjunto de parámetros a optimizar, sino con la codificación de estos. El conjunto de parámetros a optimizar en el problema de optimización se codifica en cadenas de longitud finita sobre un alfabeto finito.

- Los Algoritmos Genéticos buscan en una población de puntos, no en un solo punto.

- Los Algoritmos Genéticos usan la información de una función de coste para guiar la búsqueda, no derivadas u otro conocimiento adicional.

- Los Algoritmos Genéticos usan reglas de transición probabilísticas para pasar de una generación a otra, no determinísticas.

Estas características contrastan con las de otros procedimientos de optimización, que necesitan abundante información auxiliar para operar adecuadamente, como por ejemplo, las técnicas de gradiente. Un Algoritmo Genético, por el contrario, no necesita

ningún tipo de información auxiliar para realizar la búsqueda; son procedimientos de búsqueda ciegos. Para realizar la búsqueda, un Algoritmo Genético sólo requiere los valores de la función objetivo asociados con las correspondientes cadenas de parámetros a optimizar. Las características citadas anteriormente contribuyen a la robustez de los Algoritmos Genéticos y resultan una ventaja sobre otro tipo de algoritmos para problemas de optimización.

A pesar de los buenos resultados conseguidos, un Algoritmo Genético no resolvería de manera óptima *cualquier* problema de optimización. Esta idea está relacionada con el conocido como el teorema *no free lunch*, que establece que el rendimiento medio de un Algoritmo Genético promediado sobre todas las funciones objetivo posibles no supera al rendimiento de una búsqueda aleatoria. Este teorema propugna una especialización de los Algoritmos Genéticos para cada tipo de problema, mediante diferentes adaptaciones.

## 3.3 Descripción general del funcionamiento de un Algoritmo Genético.

Los individuos son el elemento básico de los Algoritmos Genéticos, y representan las posibles soluciones a un problema que se desea resolver. Las características de los individuos se codifican de tal forma que se puedan diferenciar, separar y combinar con las de otros individuos. La unión (en forma de lista) de estas características se denomina cromosoma.

La definición de la estructura del cromosoma es uno de los procesos críticos de cualquier exploración evolutiva, ya que debe codificar de forma clara y sencilla las características que describen a los individuos del espacio de búsqueda. Este esquema debe ser sencillo, para poder trabajar fácilmente con él, y lo más completo posible.

La función objetivo asociada a un individuo evalúa su capacidad y/o eficiencia para resolver de forma satisfactoria un problema. Para realizar esta evaluación, la función objetivo decodifica las características descritas en el cromosoma y cuantifica la

solución del problema obtenida con estas características. En este paso es donde se necesita el máximo de recursos computacionales, ya que esta función se evaluará para todos los individuos durante todas las generaciones. Por este motivo, debe estar bien diseñada y ser lo más eficiente posible (tanto al evaluar la calidad del individuo como para ser calculada).

### 3.3.1 Codificación y función objetivo de un Algoritmo Genético.

El funcionamiento intrínseco de un Algoritmo Genético obliga a que los individuos se codifiquen como cadenas de elementos, generalmente de una longitud fija *l*. Tradicionalmente se emplean cadenas binarias para codificar cada individuo [Goldberg89], aunque también se pueden emplear cadenas de enteros finitos o incluso cadenas de números reales cuantificados, siempre que el alfabeto utilizado sea finito. Usualmente, un mismo problema admite varias codificaciones, y los resultados obtenidos por un Algoritmo Genético utilizando diferentes codificaciones pueden ser completamente distintos. Esto indica que hay formas de codificar los individuos de un Algoritmo Genético que son mejores que otras. En general, una codificación es mejor en tanto en cuanto simplifique los operadores genéticos. Si la codificación es mala, puede ser necesario modificar los operadores genéticos para adaptarlos a los individuos, lo que suele ocasionar un mayor coste computacional del algoritmo y unos resultados en general pobres [Lai96].

Directamente relacionada con la codificación establecida en el Algoritmo Genético, está su función objetivo. Tradicionalmente, el objetivo de un Algoritmo Genético es maximizar (o minimizar) una función $F(X_1,X_2,...X_l)$. Se puede considerar el problema de optimización como una "caja negra", donde se introduce cada individuo y se obtiene una medida de lo bueno o malo que es ese individuo. Esta medida sería la función objetivo $F$. Existen interacciones entre los parámetros $X_i$ de la función $F$ que deben considerarse para maximizar la salida de la "caja negra". A este efecto de interacción entre variables se le conoce como epistasis del problema [Davidor91, Heckendorn99]

### 3.3.2 Operadores genéticos.

En los Algoritmos Genéticos existen tres operadores básicos para el manejo de los individuos entre generación y generación. Estos son:

**Copia.** Este operador es uno a uno, y se encarga de replicar un individuo de una generación en la siguiente. Se usa en la variante del elitismo, en la cual se copia el mejor individuo de una generación en la siguiente. De esa manera se garantiza que la mejor solución encontrada hasta ese momento persista en la siguiente generación.

**Cruce.** Este operador es dos a uno. Con este operador se seleccionan dos individuos, a los que se denominan padres, y por medio de la combinación de segmentos de sus cromosomas, se construye un nuevo individuo, al cual se denomina heredero, que contendrá características mezcladas de ambos padres y formará parte de la población de la siguiente generación. También es frecuente que este operador sea dos a dos, y el segundo heredero se construye con la información no utilizada de los padres. Este es el modelo de la reproducción sexual, y permite la creación de nuevos individuos a partir de la información explorada con anterioridad.

Los cromosomas se pueden combinar de diversas maneras. Un método que ha demostrado ser eficiente es el que se denomina cruce multipunto, que consiste en seleccionar varios puntos de corte [Jong92]. Para construir el cromosoma de un heredero se escoge un padre, y se va tomando su información genética hasta encontrar un punto de corte, a partir del cual se escoge al segundo padre para seguir con el mismo proceso, regresando al primer padre al encontrar el siguiente punto de corte, y así sucesivamente. Un ejemplo se puede observar en la Fig.3.1, donde se han seleccionado dos puntos de corte, las posiciones 4 y 9. Para el primer heredero se escogen las características 1-3 del primer padre, 4-8 del segundo y 9-10 de nuevo del primero. Para el segundo heredero se escogen los datos que no han sido utilizados en la construcción del primero.

El operador de cruce está controlado por una probabilidad de cruce $P_c$. Usualmente se escoge estrictamente menor que la unidad, y típicamente con valores de

0.6 o 0.7. Esta probabilidad hace que no haya cambios bruscos entre una generación y otra en la composición de los individuos que la forman. El operador de cruce ha sido definido como la clave del funcionamiento de los Algoritmos Genéticos [Goldberg89], y la mayor parte de las propiedades que exhiben este tipo de algoritmos son debidas a este operador.

CROMOSOMA

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Padre1<br>Padre2 | a<br>z | b<br>y | c<br>x | d<br>w | e<br>v | f<br>u | g<br>t | h<br>s | i<br>r |
| Máscara | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 1 | 1 |
| Heredero1 | a | b | c | w | v | u | t | h | i |
| Heredero2 | z | y | x | d | e | f | g | s | r |

Fig.3.1: Operador de cruce, construcción de los herederos

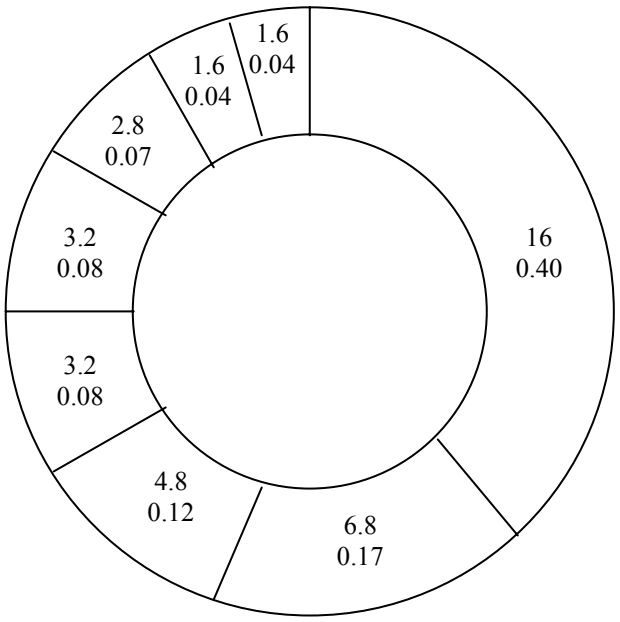
**Mutación.** Es la operación de introducir ruido en el entrenamiento genético. Este operador escoge al azar individuos de la población y modifica de forma aleatoria algunas de sus características [Hinterding95]. En el caso de que la cadena que compone el algoritmo sea binaria, el bit elegido para mutar cambia de estado de 0 a 1 o viceversa. Esto puede ayudar en la búsqueda, y permite la creación de individuos con información que no existía en la primera generación. También permite mantener la riqueza de información de la población y evitar el estancamiento.

La actuación de este operador está gobernada por una probabilidad de mutación $P_m$, asociada a cada individuo de la población. Esta probabilidad es pequeña, del orden del 1%, para que la búsqueda aleatoria que propugna la mutación no bloquee la búsqueda realizada mediante el operador de cruce.

### 3.3.3 Proceso de selección.

El proceso de selección es quizá, junto con el diseño del cromosoma, el proceso más crítico de los Algoritmos Genéticos. El proceso de selección se encarga de escoger los individuos que van a tomar parte en el proceso de reproducción. Esta selección se basa en la evaluación de la función objetivo de los individuos. Los individuos con mayor probabilidad de entrar en la reproducción serán aquellos cuya función objetivo sea mayor, aunque es importante que los individuos con una menor evaluación también tengan una cierta probabilidad de intervenir en este proceso.

El método más común y más simple es el de la ruleta [Goldberg89]. Debido a que este método ofrece buenos resultados para el problema abordado en esta tesis, se utilizará este método. En él, se divide una circunferencia en particiones de longitud proporcional a la función objetivo, como se puede observar en la Fig.3.2. Se generan números aleatorios en el rango [0,1) y se observa a que partición de la circunferencia pertenece. El individuo correspondiente a esa partición es seleccionado para reproducirse con un segundo individuo que se escoge de la misma forma. Este proceso se repite hasta completar los individuos de la siguiente generación.




| Individuo | Eval. | % |
|---|---|---|
| 1 | 16 | 40 |
| 2 | 6.8 | 17 |
| 3 | 4.8 | 12 |
| 4 | 3.2 | 8 |
| 5 | 3.2 | 8 |
| 6 | 2.8 | 7 |
| 7 | 1.6 | 4 |
| 8 | 1.6 | 4 |

Fig.3.2: Selección por el método de la ruleta

El inconveniente de este procedimiento se presenta si aparece lo que se denomina un "super-individuo", una solución cuyo valor de la función objetivo es muy alto en comparación con los demás individuos de su generación; su partición en la circunferencia prácticamente abarcará la unidad, y en el momento de escoger a los padres de la siguiente generación su probabilidad de reproducción será próxima a 1 y, como consecuencia, colmará la población de réplicas suyas, producto de combinarse consigo mismo repetidas veces.

Existen múltiples técnicas para evitar que se impongan los "super-individuos". Entre otras, limitar el número de veces que un individuo interviene en el proceso de reproducción, limitar a un cierto número la existencia de individuos repetidos en la siguiente generación, o modificar el esquema de particionar la circunferencia con la construcción de funciones crecientes con la función objetivo, pero que dividan la circunferencia de manera menos directa.

### 3.3.4 Secuencia de los Algoritmos Genéticos.

Para cualquier problema de optimización a resolver, la secuencia de los Algoritmos Genéticos es muy sencilla, y consta de los siguientes pasos:

1. Creación de la primera población de individuos.
2. n = 0.
3. Evaluación de los individuos de la población.
4. Selección de los candidatos a ser padres.
5. Creación de la nueva población con el cruce de individuos.
6. Mutación de algunos individuos.
7. n = n + 1.
8. Parar si n es mayor que la máxima generación o la función objetivo ha alcanzado su máximo valor.
9. Ir al paso 2.

En el momento de adaptar el Algoritmo Genético para resolver un problema o realizar una optimización determinada, se definen la estructura del cromosoma, la

función objetivo y la política de selección de los individuos. Es preferible escoger una población grande (valores típicos de 100 a 200 individuos, o incluso más, dependiendo del tamaño del espacio de búsqueda) para asegurar la riqueza de individuos en ésta, y el número de generaciones establecerá el grado de especialización de los individuos para solucionar el problema planteado.

Por último, hay que resaltar que un Algoritmo Genético tiene una gran variedad de parámetros aleatorios (punto de corte en los cromosomas de los padres para crear nuevos individuos, probabilidad de cruce y mutación no es la unidad,…), que hacen que la solución pueda no ser igual para dos realizaciones del mismo Algoritmo con los mismos datos de entrada. Como se recordará para el propósito de esta tesis, el algoritmo de reasignación dinámica de *slots* TSR se encuentra distribuido en todos los terminales de una red con capacidad TSR, es decir, este algoritmo reside en cada terminal de la red. Esto quiere decir que, con un Algoritmo Genético, cada terminal participante en la red puede obtener distintas asignaciones de bloques de *slots* con los mismos datos de peticiones de *slots* de todos los participantes. Por ello, en esta tesis se propondrá una variante de un Algoritmo Genético, en la que todos los parámetros aleatorios queden fijados a un mismo valor. Con esto se consigue que el Algoritmo Genético siempre obtenga la misma solución para todos los terminales de la red.

# Capítulo 4 Teoría de Juegos

## 4.1 Introducción.

La Teoría de Juegos modela los problemas de enfrentamiento de intereses entre dos o más individuos como un juego de interacción entre los participantes y, a través del estudio de las acciones que realizan los individuos que interactúan, sugiere soluciones razonables para distintas clases de juegos y examina sus propiedades. El uso de Algoritmos Genéticos para la búsqueda de estrategias con las cuales participar en un problema modelado con la Teoría de Juegos ha sido ampliamente probado, con resultados satisfactorios.

El estudio de la Teoría de Juegos comienza en la primera mitad del siglo XX con los trabajos realizados por el matemático húngaro John von Neumann, autor, junto con el economista de Princeton Oskar Morgenstern, del libro "*Theory of Games and Economic Behavior*" [Neumann44]. A finales de los años 50 la Teoría de Juegos perdió interés para volver a estar en auge a principios de la década de los 80 con trabajos como los de Robert Axelrod [Axelrod84], los estudios sobre juegos de guerra en la Guerra del Golfo en 1991 [Zweben94] y el reconocimiento con el Premio Nobel en Economía de 1994 otorgado a John F. Nash, John C. Harsanyi y Reinhard Selten.

Un juego es una descripción de una interacción estratégica que incluye los parámetros que definen las acciones que pueden ejecutar los participantes y los intereses de éstos, pero no especifica las acciones que los participantes van a llevar a cabo. La solución es una descripción sistemática de los resultados que van emergiendo. La Teoría

de Juegos sugiere soluciones razonables para diferentes clases de juegos y examina sus propiedades.

La Teoría de Juegos es un conjunto de herramientas analíticas diseñadas para ayudar a entender los fenómenos observados en relación con la toma de decisiones por parte de individuos que interactúan. El principio para este análisis asume que los individuos en acción pretenden alcanzar unos objetivos definidos por un ente externo, y tienen en cuenta (a la hora de comportarse) tanto el conocimiento propio como el comportamiento que esperan de los demás individuos.

Los modelos usados en la Teoría de Juegos son representaciones abstractas de distintos tipos de problemas de la vida real. Estas representaciones permiten estudiar un amplio número de situaciones. Por ejemplo, la Teoría del Equilibrio de Nash puede emplearse para el estudio de competiciones oligopolísticas y políticas.

En todo modelo teórico de juego, la entidad básica es el jugador, que puede ser interpretado como un individuo o grupo de individuos tomando decisiones. Una vez que se ha definido un conjunto de jugadores, se puede distinguir entre dos tipos de modelos de juegos: el primero está compuesto por un conjunto de jugadores cuyas acciones se estudian y evalúan individualmente; el segundo está compuesto por un conjunto de jugadores cuyo estudio y evaluación se realiza analizando el comportamiento y los resultados globales del grupo. Se suele denominar no-cooperativo al primero de los modelos, mientras que el segundo recibe el nombre de cooperativo.

Para clarificar la naturaleza de la Teoría de Juegos, se puede contrastar con la teoría del equilibrio competitivo empleada en economía. El razonamiento de la Teoría de Juegos analiza los intentos de cada individuo para obtener información acerca del comportamiento de los otros individuos antes de tomar una decisión, mientras que el razonamiento competitivo asume que cada agente está interesado únicamente en algunos parámetros definidos por el entorno en el que se desenvuelve, aunque dichos parámetros vengan determinados por las acciones de todos los agentes.

## 4.2 El comportamiento racional en la Teoría de Juegos.

En los modelos usados en la Teoría de Juegos se asume que cada participante es "racional", en el sentido que es consciente de sus alternativas, crea expectativas acerca de lo que va a suceder, tiene claras preferencias y determina su acción deliberadamente después de algún proceso de optimización. Así, un modelo de selección racional consta de los siguientes elementos:

- Un conjunto *A* de acciones entre las cuales el individuo decide elegir una.

- Un conjunto *C* de posibles consecuencias de estas acciones.

- Una función de consecuencia $g:A \rightarrow C$, que asocia las consecuencias con sus respectivas acciones.

- Una relación de preferencia (una relación transitiva, reflexiva y binaria) $\tau$ en el conjunto *C*.

En algunos casos, las preferencias del individuo vienen especificadas a partir de una función de utilidad $U:C \rightarrow \Re$, que define la relación de preferencia $\tau$ mediante la condición $x \, \tau \, y$ si y solo si $U(x) \geq U(y)$.

Tomando cualquier subconjunto *B* de acciones que son posibles en algún caso particular ($B \subseteq A$), un individuo racional escogerá, si es posible, la acción $a^*$, óptima en el sentido de que $g(a^*) \, \tau \, g(a)$ para todo $a \in B$, lo que resuelve el problema $max_{a \in B} U(g(a))$. Dicha condición es válida siempre y cuando el individuo use la misma relación de preferencia cuando escoge distintos subconjuntos *B*.

En los modelos estudiados en la Teoría de Juegos, los individuos toman siempre decisiones en ciertas condiciones de incertidumbre, que pueden ser:

- Incertidumbre acerca de los parámetros del entorno.

- Información incompleta acerca de los sucesos que acontecen en el juego.

- Incertidumbre acerca de las acciones de los otros jugadores que no son de tipo determinístico.

- Incertidumbre acerca del razonamiento de los otros jugadores.

El individuo en condiciones de incertidumbre, como en toda la Teoría de Juegos, usa las teorías de von Neumann y Morgenstern [Neumann44], que dicen:

- Si la función de consecuencia es estocástica y conocida para un individuo (esto es, para cada $a \in A$ el resultado $g(a)$ es un resultado probabilístico en el conjunto $C$), se supone que el individuo se comportará maximizando el valor esperado de la función que asigna un valor a cada consecuencia.

- Si la conexión estocástica entre las acciones y las consecuencias no existe, el individuo se comportará como si tuviese en mente la función de distribución de probabilidad que determina las consecuencias de cada acción. En este caso, se asume que el individuo se comporta como si tuviese en memoria el conjunto $\Omega$ de posibles estados del juego (siendo cada uno de los estados la descripción de todas las características relevantes de los jugadores), la medida de probabilidad sobre $\Omega$, la función $g:A \times \Omega \rightarrow C$, y la función de utilidad $U:C \rightarrow \Re$. El individuo escogerá una acción $a \in A$ y un estado $\omega \in \Omega$ para maximizar el valor esperado de $U(g(a,\omega))$ con respecto a la función de distribución de probabilidad.

## 4.3 Dilema del Prisionero Iterado.

El Dilema del Prisionero es un juego estratégico simétrico en el cual dos prisioneros son interrogados. Cada prisionero tiene dos opciones, que consisten en delatar a su compañero o cooperar con él y mantenerse callado. En la Fig.4.1 se puede observar la descripción del juego. Los prisioneros tienen que escoger una de estas dos acciones sin saber cual será la acción llevada a cabo por su compañero. Si los dos

deciden cooperar mutuamente, obtienen una recompensa por mantenerse callados *R*. Si uno decide cooperar y el otro decide delatar, el que ha delatado recibe la recompensa por la cual ha caido en la tentación de delatar *T*, y su compañero recibe el castigo por ingenuo *S*. Si los dos se delatan mutuamente, recibirán igualmente el castigo *P* de mayor puntuación porque comparten el castigo del entorno.

| | | Jugador de las Columnas: Cooperar | Jugador de las Columnas: Delatar |
|---|---|---|---|
| Jugador de las Filas | Cooperar | $R = 3, R = 3$ Recompensa por la mutua cooperación | $S = 0, T = 5$ Puntaje por embaucado, y tentación para delatar |
| Jugador de las Filas | Delatar | $T = 5, S = 0$ Tentación para delatar y puntaje para embaucar | $P = 1, P = 1$ Castigo por delatarse mutuamente |

Fig.4.1: Descripción del Juego del Dilema del Prisionero

Este juego sencillo puede modelar perfectamente situaciones reales tales como enfrentamientos en un duopolio mediante políticas de mercado, o enfrentamientos entre dos naciones rivales.

La solución a este juego se obtiene observando los posibles perfiles de acciones, a partir de los que podemos concluir lo siguiente: si se decide cooperar siempre, el oponente puede decidirse a cooperar, ante lo cual ambos obtienen una recompensa mediana *R*; si por el contrario, el oponente decide delatar, entonces se obtendrá el máximo castigo existente *S*. Por otra parte, si se decide delatar siempre, se puede obtener la máxima recompensa *T*, o una mínima ganancia *P*. Desde el análisis de juego no cooperativo, en el cual sólo importa el beneficio de cada uno de los jugadores por

separado, la mejor estrategia es la de "riesgo máximo menor", que en este caso es delatar.

Para modelar mejor la realidad con este juego, se define el Dilema del Prisionero Iterado, en el cual los oponentes se enfrentan varias veces en lugar de una sola. El conocimiento del número de veces que van a enfrentarse los individuos es también importante: si los oponentes conocen cuantas veces se van a enfrentar, entonces pueden hacer el análisis descrito en el párrafo anterior para la última jugada y de manera recurrente llegar al principio del juego. En este caso, de nuevo, la mejor estrategia es siempre delatar.

El verdadero dilema se presenta cuando no se conoce el número de veces que van a enfrentarse. En cada iteración, el jugador sólo sabe lo que sucedió en las jugadas anteriores, tanto suyas como de su contrincante, pero desconoce el número de veces que le quedan por enfrentarse a él. Adicionalmente, el jugador sabe que delatar siempre es una buena estrategia, porque su oponente nunca se podrá llevar la mejor recompensa ($T$), pero lo más seguro es que ambos obtengan resultados muy pobres. Lo mejor sería arriesgarse a cooperar y que el oponente hiciera lo mismo. De esa forma, los beneficios para los dos jugadores serían mayores. Es aquí cuando se plantea el dilema: arriesgarse a cooperar y tener la posibilidad de que los dos jugadores obtengan buenos beneficios, o delatar siempre y obtener beneficios pobres. De esta forma se demuestra que no existe una estrategia óptima para resolver este juego, ya que la mejor estrategia vendrá definida por la estrategia del oponente.

El Dilema del Prisionero Iterado fomenta la cooperación entre contrincantes para obtener de manera conjunta el mayor beneficio. Para que surja la cooperación es importante que las recompensas y castigos mantengan unas relaciones bien definidas. La primera relación que se debe mantener es de orden entre las diferentes recompensas: $T > R > P > S$. Si se altera esta relación, la definición del juego ha variado y ya no corresponde al Dilema del Prisionero. La segunda relación, $2R > T + P$, establece que la recompensa por cooperar debe ser mayor que el promedio de las recompensas obtenidas al delatar al oponente. De esta manera se evita que los jugadores que lleguen al equilibrio de delatar y cooperar de manera alternada obtengan mayores beneficios que los jugadores que siempre cooperan.

Determinado comportamiento social surgido durante la Primera Guerra Mundial [Axelrod84] constituye un claro ejemplo de la presencia del modelo del Dilema del Prisionero Iterado en la vida real. En esta contienda, y en un entorno tan adverso como fue la guerra de trincheras, surgió un esquema de cooperación denominado "vive y deja vivir", por el cual los contendientes procuraban evitar cualquier tipo de enfrentamiento, cooperando mutuamente (al no agredirse). En este enfrentamiento se pueden observar las siguientes características anteriormente definidas para el Dilema del Prisionero Iterado:

- La relación entre las distintas recompensas se pueden observar de la siguiente manera:

    - Delatar, Delatar: la recompensa obtenida por el combate era baja, porque al causar víctimas en el bando contrario se debilitaba al enemigo, lo que resultaba conveniente para el conflicto, pero también se producían bajas en la misma medida en el ejército propio.

    - Cooperar, Delatar: en caso de un ataque unilateral, siempre era conveniente ser el agresor y no la víctima, ya que de esta manera se debilitaba al enemigo, sin que las bajas propias fueran considerables.

    - Cooperar, Cooperar: si ninguno de los dos cometía agresiones, los soldados podían sobrevivir a la contienda sin ningún problema, salvo aquellos derivados de la desaprobación de su comportamiento por parte de los mandos militares.

- La relación de orden se cumple al ser preferible siempre atacar unilateralmente, ya que se debilitaba al enemigo y la guerra podía acabar antes. Una batalla entre los dos ejércitos era preferible a que el enemigo atacase sin ningún tipo de respuesta. Pero, con respecto a estas últimas alternativas, siempre era mejor mantenerse sin realizar ningún tipo de agresión.

- Se cumple la relación de magnitud de las recompensas $2R > T + P$. De no cumplirse esta relación, habría sido más efectivo realizar ataques unilaterales de manera alternada, causando un gran número de bajas en el bando contrario.

- El número de "iteraciones", en este caso escaramuzas, era desconocido para los participantes.

En 1980 Robert Axelrod planteó un torneo para buscar la mejor estrategia para este juego, y convocó a investigadores de múltiples disciplinas como matemáticas, economía, ciencias políticas, etc. Los participantes enviaron sus estrategias, y los resultados fueron comunicados a la vez que se convocaba a un segundo torneo para que los participantes incluyesen modificaciones y mejoras en sus estrategias (los resultados de ambos torneos pueden encontrarse en [Axelrod84]).

Lo importante de estos dos torneos fue la selección de la mejor estrategia del concurso, que fue la misma en ambos casos. Esta estrategia, denominada TIT FOR TAT, consiste en cooperar en la primera jugada, y en las siguientes ejecutar la misma acción que el contrincante en la iteración anterior. La estrategia TIT FOR TAT no obtuvo siempre los mejores resultados, pero fue la mejor en media.

A partir de las conclusiones de este torneo se pueden inferir algunas de las características básicas que deben cumplir las buenas estrategias (aquellas que se dan en TIT FOR TAT):

- No hay que ser envidioso.

- No hay que delatar primero.

- Hay que ser recíproco tanto cooperando como delatando.

- No hay que ser demasiado listo.

## 4.4 Generalización del Dilema del Prisionero Iterado para N jugadores.

Para poder utilizar este esquema en esta tesis, es necesario generalizarlo para N participantes. Existen varias generalizaciones para N jugadores del Dilema del Prisionero Iterado clásico de 2 jugadores. Aquí se considerará la generalización debida a Schelling [Schelling78]. El número de jugadores será fijo (N). En cada jugada, cada jugador debe elegir entre cooperar o delatar. La puntuación para cada jugada dependerá de lo que se elija, así como del número total de jugadores que cooperan en esa jugada. Por consiguiente, el juego es definido por dos funciones: puntuación para los jugadores que delatan D(i) y puntuación para los jugadores que cooperan C(i), donde i es el número de jugadores que cooperan, $0 \leq i \leq N$ (realmente, D(N) y C(0) no necesitan ser definidas). Se puede definir una gran variedad de juegos definiendo estas dos funciones de diversas maneras. Se considerará que un juego es una generalización del Dilema del Prisionero si satisface las siguientes propiedades:

- $D(i - 1) > C(i)$ para todo i.

- $C(N) > D(0)$.

Así, sea cual sea el número de jugadores que cooperen, cualquier jugador tendrá mejor puntuación si delata. Pero si todos cooperan tendrán mejor puntuación que si todos delatan.

En el caso que nos ocupa, se considerarán las funciones C y D lineales con respecto al número de jugadores que cooperan. Así, el juego queda definido por el número de jugadores N y la pendiente y punto de corte con el eje x de las dos funciones de puntuación.

Para este tipo de juegos, la decisión en cada movimiento va a depender de lo que haga el resto de jugadores en el movimiento anterior. Por ello, cada estrategia se codifica como un cromosoma con N+1 posiciones [Bankes94]. Las primeras N posiciones indican las decisiones dependiendo del anterior movimiento: la posición j (j

= 0..N-1) indica la decisión cuando j del resto de jugadores cooperan en el anterior movimiento. La última posición de la estrategia indica el primer movimiento del jugador en el juego.

# Parte II

# Propuestas de Mejora del Algoritmo TSR

# Capítulo 5 Propuestas basadas en Técnicas de Optimización Combinatoria

## 5.1 Introducción.

En este capítulo se presentan las propuestas basadas en Técnicas de Optimización Combinatoria que se proponen estudiar en esta tesis para la mejora del algoritmo TSR de reasignación dinámica de *slots* actual, en el caso de que la demanda total de *slots* exceda de la capacidad disponible. Con ello, se pretende eliminar los casos de conflicto en la asignación y minimizar el valor medio de la cola máxima de mensajes no transmitidos en cada terminal participante, es decir, que para un número de *reallocation periods* dado, las colas de mensajes no transmitidos en todos los participantes en una red se mantengan lo más pequeñas posibles. Si $q_i(t)$ es la cola de mensajes para el terminal $i$ en el *reallocation period t*, el objetivo es obtener nuevos algoritmos TSR que calculen el valor de la máxima cola de mensajes en cada *reallocation period* como $max(q_i(t))$ ($i$ = 1 ... Número de terminales) y obtengan el valor medio de esos valores para todos los *reallocation periods*. La forma de alcanzar este objetivo es minimizar el número de bloques de *time slots* que se dejan de asignar a todos los terminales participantes en la red.

Para ello, se propone una nueva estrategia: a cada terminal sordo se le asigna el número de bloques de *slots* que solicita, y el problema consiste en la asignación del resto de bloques de *slots* al resto de terminales participantes, eliminando el orden de

prioridad. Esto se puede realizar debido a que la prioridad es sólo una forma de ordenar los terminales para la asignación de los bloques de *slots*, pero no es necesario.

Para resolver el problema, se han planteado diversas propuestas: una propuesta basada en el método de optimización de programación entera Branch and Bound, una propuesta basada en Redes Neuronales de Hopfield, un algoritmo heurístico y un Algoritmo Genético, y se ha realizado un estudio comparativo entre ellas en referencia a las soluciones obtenidas y al grado de complejidad de cada una de ellas (dependiendo de la potencia de cálculo y la velocidad de convergencia a una solución óptima).

## 5.2 Método *Branch and Bound* para TSR.

Como se ha comentado en el apartado anterior, el objetivo propuesto es minimizar la cola máxima de mensajes de todos los terminales no sordos participantes en cada *reallocation period*, es decir, minimizar el número de bloques de *slots* que se dejan de asignar a los participantes en el caso de que la suma de las demandas de bloques de *time slots* de todos los terminales no sordos sea mayor que el número de bloques de *slots* disponibles.

Para el terminal $i$, el número de bloques de *slots* que se dejan asignar es $Rq_i - X_i$, donde $Rq_i$ es el número de bloques de *slots* que solicita el terminal y $X_i$ es el número de bloques de *slots* que se le asigna.

Si $Rq$ es la suma de los bloques de *slots* que solicitan todos los terminales no sordos, $P_{nd}$ es el número de bloques de *slots* disponibles en modo de acceso TSR para esos terminales y $Rq > P_{nd}$, el número total de bloques de *slots* que se dejan de asignar a todos los terminales no sordos es $Rq - P_{nd}$. Pues bien, la cola máxima de mensajes más pequeña se aproximará al valor medio del número de bloques de *slots* que se dejan de asignar a cada terminal participante.

Si $N_{nd}$ es el número de terminales no sordos participantes en el *pool* de *slots* disponible para esos terminales, entonces

$$\bar{x} = \frac{\sum_{i=1}^{Nnd} Rq_i - P_{nd}}{N_{nd}} \tag{5.1}$$

es el valor medio del número de bloques de *time slots* que se dejan de asignar a cada participante. Así pues, el número de bloques de *time slots* que se dejan de asignar al terminal $i$, $Rq_i - X_i$, debe estar próximo al número dado por la ecuación (5.1). El objetivo de nuestro algoritmo es minimizar la función de coste

$$f(x) = \sum_{i=1}^{Nnd} \left[ \left( Rq_i - X_i - \bar{x} \right)^2 + \left( Rq_i - X_i \right) \right] \tag{5.2}$$

Operando sobre la función anterior se obtiene

$$f(x) = \sum_{i=1}^{Nnd} \left( X_i^{\,2} - A_i X_i + B_i \right) \tag{5.3}$$

donde

$$\begin{aligned} A_i &= 2\left( Rq_i - \bar{x} \right) + 1 \\ B_i &= \left( Rq_i - \bar{x} \right)^2 + Rq_i \end{aligned} \tag{5.4}$$

Las restricciones a nuestro problema son las siguientes: la suma del número de bloques de *slots* asignados a todos los terminales participantes debe ser igual al número de bloques de *slots* disponibles para TSR y, para cada terminal participante, el número de bloques de *slots* asignados debe ser menor o igual que el número de bloques de *slots* que solicita.

Por lo tanto, el problema a resolver consiste en minimizar la función objetivo (5.3) con las siguientes restricciones:

$$\sum_{i=1}^{Nnd} X_i = P_{nd}$$

$$X_i \leq Rq_i \qquad i = 1....N_{nd} \tag{5.5}$$

Como las variables $X_i$ deben ser enteras, es un problema de optimización de programación entera, donde la función objetivo *f(x)* es cuadrática y las restricciones *Ci(x)* son lineales. Para resolver este problema se ha aplicando el método de programación entera de *Branch and Bound* [Fletcher87], ya que es un método con una amplia generalidad y una eficiencia razonable.

Este método utiliza el cálculo de un límite inferior de una instancia del problema a resolver y la división de la región de posibles valores de las variables *x* para crear subproblemas más pequeños. Se debería también poder calcular un límite superior para algunas instancias del problema.

El método comienza considerando el problema original con la región completa de posibles valores de las variables *x*, y se aplican los procedimientos de límite inferior y límite superior a este problema. Si los límites coinciden, quiere decir que se ha encontrado la solución óptima y termina el procedimiento. Si no, se divide la región inicial en dos o más regiones diferentes, de forma que, entre todas ellas, cubran la región original; idealmente, estos subproblemas dividen la región original. Se aplica el algoritmo a estos subproblemas de forma recursiva, generando un árbol de subproblemas. Si se encuentra una solución óptima para un subproblema, es una solución posible para el problema completo, pero no necesariamente es una solución global. Ya que es una solución posible, se puede utilizar para cortar el resto del árbol: si el límite inferior para un nodo excede la mejor solución posible conocida, no puede existir una solución óptima global en el subespacio de la región representada por el nodo. Por lo tanto, se puede eliminar el nodo. La búsqueda continúa hasta que todos los nodos han sido resueltos o cortados, o hasta que se encuentre un umbral específico entre la mejor solución encontrada y los límites inferiores de todos los problemas que no se han resuelto.

## 5.3 Red Neuronal de Hopfield para TSR.

El objetivo de nuestro algoritmo es minimizar la máxima cola de mensajes que se dejan de transmitir para todos los terminales en cada *reallocation period*. Como se ha comentado en el apartado anterior, la cola máxima de mensajes más pequeña se aproximará al valor medio del número de bloques de *slots* que se dejan de asignar a cada terminal participante, que viene dado por la ecuación (5.1). Pues bien, para cumplir el objetivo deseado se asignará un mínimo de bloques de *slots* a cada terminal, que será el número de bloques de *slots* que solicita menos el valor dado por la ecuación anterior. Así pues, las restricciones a nuestro problema son las siguientes:

1. Un bloque de *time slots* no puede ser asignado a más de un terminal.

2. Todos los bloques de *time slots* deben ser asignados a los terminales.

3. El número de bloques de *time slots* asignado a cada terminal debe ser menor o igual que el número de bloques de *time slots* que solicita ese terminal.

4. El número de bloques de *time slots* asignado a cada terminal debe ser mayor o igual que el número de bloques de *time slots* que solicita ese terminal menos el valor medio redondeado hacia arriba del número de bloques de *time slots* que se dejan de asignar a cada terminal participante.

Se pueden cumplir estos requisitos utilizando una red neuronal de Hopfield binaria siguiendo la definición de Funabiki *et al.*[Funabiki97], usando primero la ecuación de movimiento en lugar de la función energía. Para ello se define una red neuronal bidimensional. La salida $V_{ij}$ de la neurona *ij* representa si el bloque de *time slots j* es asignado al terminal participante *i*. $V_{ij} = 1$ representa la asignación, y $V_{ij} = 0$ la no asignación. Para este problema se necesitan $N_{nd}$ x $P_{nd}$ neuronas, con $N_{nd}$ terminales participantes y $P_{nd}$ bloques de *time slots*. La Fig.5.1 muestra la red neuronal compuesta de 32 (4 x 8) neuronas y el estado de convergencia a la solución correspondiente a un problema con un vector de peticiones *rq* = [2 4 3 6]. El cuadrado negro indica la salida uno y el cuadrado blanco indica la salida cero. Como se puede observar, se asigna un

bloque de *slots* al primer terminal participante, dos al segundo terminal, uno al tercer terminal y cuatro al cuarto terminal. Con ello, el máximo número de bloques de *slots* que se dejan de asignar corresponden a los terminales segundo, tercero y cuarto, y es dos, que es el valor más pequeño posible.

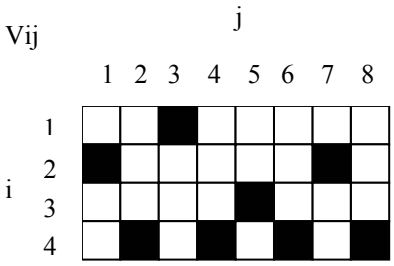



Fig.5.1: La red neuronal para un vector de peticiones *rq* = [2 4 3 6]

Con esta representación, se pueden formular las restricciones anteriores de la siguiente forma:

$$1.\quad \sum_{i=1}^{Nnd} V_{ij} \le 1 \qquad j = 1 \ldots P_{nd}$$

$$2.\quad \sum_{i=1}^{Nnd} \sum_{j=1}^{Pnd} V_{ij} = P_{nd}$$

$$3.\quad \sum_{j=1}^{Pnd} V_{ij} \le rq_i \qquad i = 1 \ldots N_{nd}$$

$$4.\quad \sum_{j=1}^{Pnd} V_{ij} \ge x_i \qquad i = 1 \ldots N_{nd}$$

donde $rq_i$ es el número de bloques de *time slots* que solicita el terminal $i$, y $x_i$ es el número de bloques de *time slots* que solicita este terminal menos el valor medio redondeado hacia arriba del número de bloques de *time slots* que se dejan de asignar a cada terminal participante en la red.

La segunda restricción es demasiado restrictiva para la convergencia de la red a una solución, así que se eliminará en la ecuación de movimiento. En su lugar, se

añadirán en dicha ecuación dos términos para incrementar el número de bloques de *time slots* asignados tanto como sea posible. Los resultados obtenidos con las simulaciones realizadas en esta tesis mostrarán que la red neuronal propuesta puede encontrar la solución global, es decir, se asignan todos los bloques de *time slots* a los terminales participantes.

Con todo ello, para nuestra red neuronal, la ecuación de movimiento que conduce el estado de la red a una solución para el problema planteado se define de la siguiente forma para la neurona *ij*:

$$\frac{dU_{ij}}{dt} = -A\left(\sum_{\substack{q=1 \\ q\neq i}}^{Nnd} V_{qj}V_{ij}\right) - Bf\left(\sum_{t=1}^{Pnd} V_{it} - rq_i\right) - Cg\left(\sum_{t=1}^{Pnd} V_{it} - x_i\right) + Dh\left(\sum_{t=1}^{Pnd} V_{it} - x_i\right) + Eh\left(\sum_{q=1}^{Nnd} V_{qj} - 1\right) \tag{5.6}$$

donde $U_{ij}$ es la entrada de la neurona *ij* y *A*, *B*, *C*, *D*, *E* son coeficientes constantes.

El primer término de la ecuación representa la primera restricción en este problema. Impide que la neurona *ij* tenga salida uno si se asigna el bloque de *time slots j* a otro terminal distinto de *i*. El segundo término representa la tercera restricción. Impide que la neurona *ij* tenga salida uno si se asignan al terminal *i* más de $rq_i$ bloques de *time slots*. La función *f(x)* es "1" si $x > 0$ y "0" en el resto de los casos. El tercer término representa la cuarta restricción. Impide que la neurona *ij* tenga salida uno si se asignan al terminal *i* menos de $x_i$ bloques de *time slots*. La función *g(x)* es *x* si $x < 0$ y "0" en el resto de los casos.

Los dos últimos términos de la ecuación son los términos heurísticos llamados *hill-climbing*, siguiendo [Funabiki92a, Funabiki92b, Funabiki93, Funabiki95]. La función *h(x)* es "1" si $x < 0$ y "0" en el resto de los casos. El objetivo de estos términos es incrementar el número de bloques de *time slots* asignados tanto como sea posible.

Para la actualización de la salida de la neurona se adopta el modelo neuronal binario de McCulloch-Pitts [McCulloch43]: si $U_{ij} > 0$ entonces $V_{ij} = 1$, si no $V_{ij} = 0$.

El cálculo iterativo de la ecuación de movimiento y la función de la neurona debe terminar cuando el estado de la red neuronal satisface todas las restricciones y la función objetivo del problema, es decir, cuando todos los términos en la ecuación de movimiento son cero.

En cuanto a la relación entre las ecuaciones de este método y el método anterior de *Branch and Bound*, el tercer y el cuarto términos de la ecuación de movimiento de la red de Hopfield corresponden a la función objetivo del problema en el método de *Branch and Bound*, ya que utilizan el parámetro $x_i$, que engloba al valor medio del número de bloques de *slots* que se dejan de asignar a cada terminal participante, parámetro que viene dado por la ecuación (5.1). El resto de términos de la ecuación de movimiento, incluyendo el cuarto también, corresponden a las restricciones del problema en el método de *Branch and Bound*.

## 5.4 Algoritmo heurístico para TSR.

La red neuronal de Hopfield propuesta anteriormente es muy lenta en cuanto a convergencia a la solución global cuando el número de participantes en la red y el número de bloques de *time slots* disponibles para TSR son muy altos. Para evitar ese inconveniente, se propone un algoritmo heurístico de asignación dinámica de *time slots*. Las mejoras introducidas en este algoritmo van encaminadas a eliminar los casos de conflicto del algoritmo actual y a minimizar el valor medio de la cola de mensajes en cada terminal participante, repartiendo los bloques de *slots* de una forma más simple y equitativa.

Como ya se ha comentado en el apartado anterior, la cola máxima de mensajes más pequeña se aproximará al valor medio del número de bloques de *slots* que se dejan de asignar a cada participante. El algoritmo propuesto se basa en asignar primero a cada terminal participante el número de bloques de *slots* solicitado menos el valor medio del

número total de bloques de *slots* que se dejan de asignar a todos los participantes y después, del resto de bloques que quedan sin asignar, se asigna un bloque de *slots* mas a cada terminal. A los terminales sordos se les asigna primero el número de bloques de *slots* solicitado. Los pasos a seguir en este algoritmo son los siguientes:

a) Se asigna a cada terminal sordo el número de bloques de *slots* solicitado y se resuelve el problema de asignar un número de bloques de *slots* $P_{nd}$ a un número de participantes $N_{nd}$, donde $P_{nd}$ es el número de bloques de *slots* disponible inicialmente para TSR menos los requeridos por los terminales sordos, y $N_{nd}$ es el número de terminales no sordos.

b) Se calcula el número total de bloques de *slots* que se dejan de asignar a los terminales no sordos como la suma de todos los bloques de *slots* solicitados por estos terminales menos el número de bloques de *slots* disponibles para ellos:

$$Bq = \sum_{i=1}^{Nnd} rq_i - P_{nd} \tag{5.7}$$

c) Si la suma de todas las peticiones de bloques de *slots* por parte de todos los terminales participantes no sordos es mayor que el número total de bloques de *slots* disponibles para estos terminales:

c.1) se asigna a cada terminal participante no sordo el número de bloques de *slots* solicitado menos el valor medio redondeado hacia arriba del número total de bloques de *slots* que se dejan de asignar. Este valor viene dado por la ecuación (5.8). Si este valor es menor que cero, no se asigna ningún bloque de *slots* al terminal.

$$(rq_i)_{\text{int}} = rq_i - round\left(\frac{Bq}{N_{nd}}\right) \tag{5.8}$$

c.2) si la suma de bloques de *slots* asignados a todos los terminales participantes no sordos es menor que el número de bloques de *slots* disponibles para estos terminales:

c.2.1) se calcula el número de bloques de *slots* que quedan por asignar como:

$$P_{nd} - \sum_{i=1}^{Nnd} (rq_i)_{\text{int}} \qquad (5.9)$$

c.2.2) se asigna un bloque de *slots* más a cada terminal participante no sordo hasta completar el número de bloques de *slots* que quedan por asignar dado por la ecuación (5.9).

c.3) si la suma de bloques de *slots* asignados a todos los terminales participantes no sordos es mayor que el número de bloques de *slots* disponibles para estos terminales:

c.3.1) se calcula el número de bloques de *slots* asignados de más como:

$$\sum_{i=1}^{Nnd} (rq_i)_{\text{int}} - P_{nd} \qquad (5.10)$$

c.3.2) se va restando un bloque de *slots* a cada terminal participante no sordo, si el número de bloques de *slots* asignado a ese terminal es mayor que cero, hasta completar el número de bloques de *slots* asignados de más dado por la ecuación (5.10).

d) Si la suma de todas las peticiones de bloques de *slots* por parte de todos los terminales participantes no sordos es menor o igual que el número de bloques de *slots* disponibles para estos terminales, se asigna a cada terminal no sordo el número de bloques de *slots* solicitado.

En cuanto a la relación entre las ecuaciones de este algoritmo y los anteriores, las correspondientes hasta el paso c.1) inclusive (ecuación (5.8)) corresponden a la función objetivo del problema, ya que el algoritmo asigna los bloques de *slots* en relación al valor medio del número de bloques de *slots* que se dejan de asignar a cada terminal participante, parámetro que viene dado por la ecuación (5.1). A partir de este paso, el algoritmo asigna los bloques de *slots* hasta cumplir las restricciones.

## 5.5 Algoritmo Genético para TSR.

Finalmente, se propone resolver el problema presentado en esta tesis mediante un Algoritmo Genético. Las características propias de estos algoritmos, citadas anteriormente en el Capítulo 3 de esta tesis, contribuyen a su robustez y resultan una ventaja sobre el resto de algoritmos para problemas de optimización.

Como se ha comentado en el apartado 5.2, el problema a resolver consiste en minimizar la función objetivo dada por la ecuación (5.3):

$$f(x) = \sum_{i=1}^{Nnd} \left( X_i^{\,2} - A_i X_i + B_i \right)$$

con las restricciones dadas por las ecuaciones (5.5):

$$\sum_{i=1}^{Nnd} X_i = P_{nd}$$

$$X_i \le Rq_i \qquad i = 1 \ldots . N_{nd}$$

El objetivo de un Algoritmo Genético es maximizar una función de coste *f(x)*. Para transformar un problema de minimización como el que se desea resolver en un problema de maximización, la transformación de la función de coste más utilizada es [Goldberg89]:

$$g(x) = \begin{cases} C_{max} - f(x) & \text{si } f(x) < C_{max} \\ 0 & \text{resto} \end{cases} \qquad (5.11)$$

y el problema es ahora maximizar la función de coste *g(x)* dada por la ecuación (5.11). Se puede escoger el coeficiente $C_{max}$ de diversas formas: un coeficiente de entrada seleccionado por el usuario, el valor más grande de la función *f(x)* observado hasta ese momento, el valor más grande de la función *f(x)* en la población actual, o el valor más

grande de las últimas *k* generaciones. Quizás lo mas apropiado sería que el valor de $C_{max}$ pudiese variar dependiendo de la variación de la población en el Algoritmo.

Los Algoritmos Genéticos procesan poblaciones de individuos, posibles soluciones del problema, que se codifican como cadenas de elementos de una longitud fija. En nuestro caso, se construye una población como un *array* de individuos. Como existen $N_{nd}$ variables en nuestro problema ($X_i$, $i = 1...N_{nd}$), cada individuo contiene el fenotipo (el valor de las variables decodificadas), el genotipo (el cromosoma o cadena de *bits* de las variables concadenadas), el valor de la función objetivo con esos valores de las variables, los padres de ese individuo y el punto de corte de los cromosomas de los padres para obtener ese individuo.

En un Algoritmo Genético, se aplican los operadores genéticos a una población completa en cada generación, como se muestra en la Fig.5.2. Para implementar esta operación se utilizan dos poblaciones, una población antigua y una nueva población. La población antigua se convierte en la nueva población al aplicar los operadores de reproducción, cruce y mutación, y se pasa de la generación T a la generación siguiente T +1.

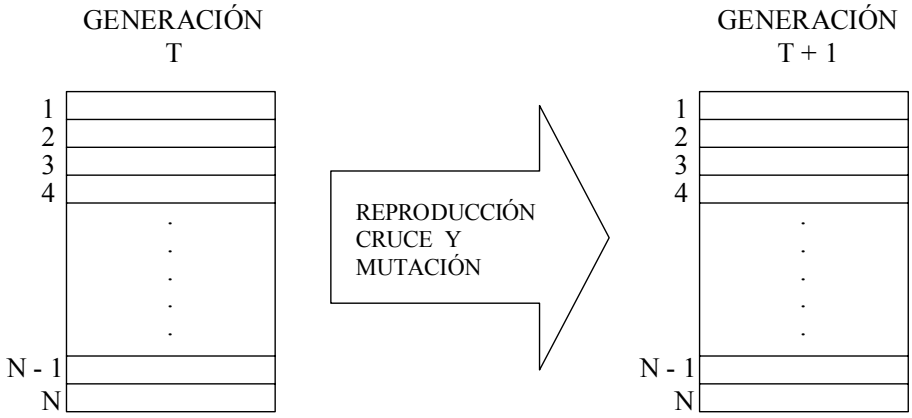



Fig.5.2: Esquema de las poblaciones utilizadas en el Algoritmo Genético

Para nuestro Algoritmo, se crea una generación inicial con un tamaño de población dado. Los individuos de esta población se seleccionan de forma aleatoria, y

son posibles soluciones del problema, es decir, son combinaciones posibles de las variables $X_i$ que cumplen las restricciones del problema. Aplicando los operadores genéticos de reproducción, cruce y mutación, se obtiene una nueva población de individuos posibles soluciones del problema que componen la generación siguiente. A esta nueva población se le aplican de nuevo los operadores genéticos y se obtiene una nueva población que compone una nueva generación. Así se continúa hasta llegar a un número máximo dado de generaciones.

En cada generación, para cada individuo de la población se decodifican los valores de la variables $X_i$ y se calcula el valor de la función objetivo con esos valores de las variables. Finalmente, se obtiene el valor máximo de la función y el valor medio de la función en esa generación. A medida que se pasa de una generación a la siguiente, el valor máximo de la función objetivo aumenta, hasta que se mantiene en un valor constante a partir de una generación dada. La solución a nuestro problema será el individuo que consigue ese valor máximo de la función objetivo.

El proceso de reproducción consiste en seleccionar parejas de individuos de una población que generen los individuos de la población de la generación siguiente. Los individuos con mayor probabilidad de entrar en la reproducción serán aquellos cuya función objetivo sea mayor. En nuestro algoritmo, se utiliza el método de la ruleta para seleccionar los individuos, que ya se ha comentado en el Capítulo 3 de esta tesis.

Mediante el operador de cruce, que se realiza con una probabilidad *pcross*, se toman los dos individuos de cada pareja seleccionada, a los que se denominan padres, y por medio de la combinación de segmentos de sus cromosomas, se construyen dos nuevos individuos, a los que se denominan herederos, que contendrán características mezcladas de ambos padres y formarán parte de la población de la siguiente generación. Para ello, se selecciona de forma aleatoria un punto de corte. Para construir el cromosoma de un heredero se escoge un padre, y se va tomando su información genética del cromosoma hasta encontrar el punto de corte, a partir del cual se escoge la información genética del cromosoma del segundo padre. Para el segundo heredero se escogen los tramos de los cromosomas de los padres que no han sido utilizados en la construcción del primero.

Finalmente, mediante el operador de mutación, que se realiza con una probabilidad *pmutation*, se modifica de forma aleatoria algunas de las características de los cromosomas de los herederos obtenidos. En nuestro caso, al ser la cadena que compone el cromosoma binaria, el *bit* elegido para mutar cambia de estado de 0 a 1 o viceversa. Esto puede ayudar en la búsqueda del algoritmo, y permite la creación de individuos con información que no existía en la primera generación.

Hasta aquí se ha descrito el algoritmo genérico. Sin embargo, como ya se ha comentado en el último párrafo del Capítulo 3, el Algoritmo Genético tiene una gran variedad de parámetros aleatorios (punto de corte en los cromosomas de los padres para crear nuevos individuos, probabilidad de cruce y mutación no es la unidad,…), que lo hacen inviable para el propósito de esta tesis, ya que la solución puede no ser igual para dos realizaciones del mismo Algoritmo con los mismos datos de entrada. Hay que recordar que el algoritmo de reasignación dinámica de *slots* TSR se encuentra distribuido en todos los terminales de una red con capacidad TSR, es decir, este algoritmo reside en cada terminal de la red. Esto quiere decir que, con un Algoritmo Genético, cada terminal participante en la red puede obtener distintas asignaciones de bloques de *slots* con los mismos datos de peticiones de *slots* de todos los participantes. Por ello, en esta tesis se propondrá una variante del Algoritmo Genético, en la que todos los parámetros aleatorios queden fijados a un mismo valor. Con esto se consigue que el Algoritmo Genético siempre obtenga la misma solución para todos los terminales de la red.

# Capítulo 6 Propuestas basadas en Técnicas Predictivas y Teoría de Juegos

## 6.1 Introducción.

En este capítulo se presentan las propuestas basadas en Técnicas Predictivas y Teoría de Juegos que se proponen estudiar en esta tesis para la mejora de la reasignación dinámica de *slots* actual, en el caso de que la demanda total de *slots* por parte de todos los terminales participantes en la red sea menor que la capacidad de *slots* disponible para el modo de acceso TSR. Cuando esto ocurre, a cada terminal se le asigna el número de bloques de *time slots* que solicita y no existe ningún conflicto, pero quedan bloques de *slots* libres, sin asignar.

Como se ha comentado en el Capítulo 1 de esta tesis, en cada *reallocation period* cada terminal envía un mensaje indicando su demanda de *slots* para transmitir sus mensajes en el siguiente *reallocation period*. Este mensaje se envía en el primer *time slot* asignado para transmitir en el *reallocation period*. Después, en los siguientes *slots*, transmite sus mensajes. Así pues, si el primer *slot* se encuentra al comienzo del *reallocation period*, las probabilidades de conocer la demanda de *slots* para el siguiente *reallocation period* de forma correcta serán más pequeñas que si el primer *slot* se encuentra al final del *reallocation period*. Además, los bloques de *slots* asignados a un terminal para la transmisión de sus mensajes en el siguiente *reallocation period* se encuentran espaciados en ese periodo, por lo cual, si se asignaran más bloques de *slots*

al terminal, podría empezar a transmitir antes sus mensajes en el *reallocation period*. Con las propuestas presentadas en este capítulo se pretende predecir esta demanda de *slots* en función de las necesidades en los últimos *reallocation periods*, asignando al terminal parte de los bloques de *slots* que quedan libres en la red, además de los demandados por él. Con ello se pretende minimizar el tiempo de transmisión de los mensajes del terminal desde que se demandan los *slots* hasta que se transmiten, ya que, al asignarle más bloques de *slots*, puede transmitir antes en el siguiente *reallocation period*.

Para ello, se han planteado dos propuestas: una propuesta basada en Técnicas Predictivas, que es un método autoregresivo (AR), y una propuesta basada en Teoría de Juegos, que es el Dilema del Prisionero Iterado para N jugadores, y se ha realizado un estudio comparativo entre ellas y el algoritmo de asignación actual en referencia al tiempo de transmisión de los mensajes del terminal.

## 6.2 Método Autoregresivo (AR) para TSR.

Como se ha comentado en el apartado anterior, el objetivo propuesto es poder predecir la demanda de *slots* de cada terminal en función de las necesidades en los últimos *reallocation periods*, asignando a cada terminal parte de los bloques de *slots* que quedan libres. Con ello se pretende minimizar el tiempo de transmisión de los mensajes de cada terminal desde que se demandan los *slots* en un *reallocation period* hasta que se transmiten en el siguiente *reallocation period*, ya que, si se le asignan más bloques de *slots*, puede transmitir antes los mensajes en el siguiente *reallocation period*.

Unos de los métodos más usados en técnicas de predicción, debido a sus buenas prestaciones, son los métodos autoregresivos (AR). Las técnicas de predicción lineal y los métodos AR están fuertemente relacionados [Akay94]. Si un proceso $y(n)$ es un proceso AR, se pueden estimar los parámetros AR en base al método de predicción lineal mostrado en la Fig.6.1, donde la potencia del error de predicción, $e(n)^2$, es la potencia de la excitación de entrada (ruido). Los coeficientes de predicción son los parámetros AR. El orden del proceso AR y del predictor lineal es el mismo.

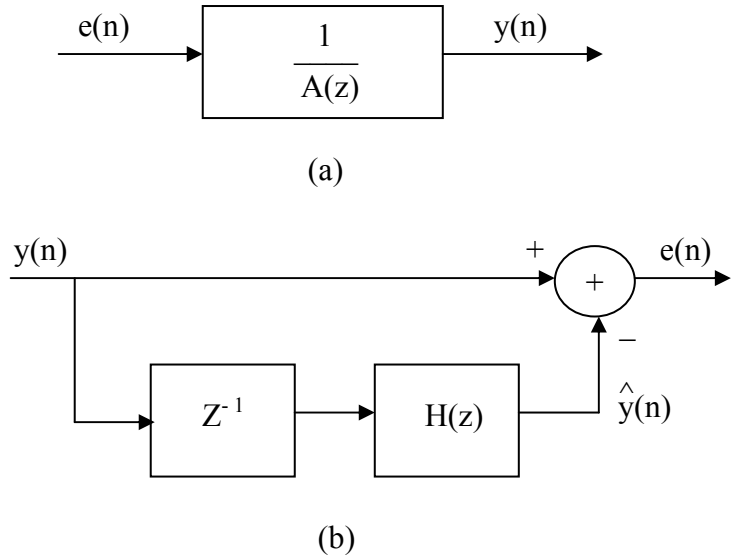



Fig.6.1: (a) Proceso AR. (b) Predictor lineal

La función de transferencia del proceso AR desde la salida *y(n)* (proceso autoregresivo) hasta la entrada *e(n)* (proceso ruido blanco) viene dada por la ecuación:

$$A(z) = 1 - z^{-1}H(z) \tag{6.1}$$

o, desarrollando la ecuación:

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots\cdots + a_M z^{-M} \tag{6.2}$$

donde *H(z)* representa el filtro Wiener y *M* representa el orden del filtro AR. Se puede escribir el proceso AR en el dominio *z* como:

$$E(z) = A(z)Y(z) \tag{6.3}$$

y en el dominio del tiempo como:

$$e(n) = \sum_{m=0}^{\infty} a_m(n)y(n-m) \tag{6.4}$$

donde *e(n)* es el error de estimación.

Los métodos AR son los más utilizados para estimar la función densidad espectral de potencia asociada a una señal discreta. Cada muestra de una señal se puede expresar como una combinación lineal de muestras previas y una señal de error *e(n)*. Se asume que esta señal error es independiente de las muestras previas. La entrada *y(n)* se puede predecir como la combinación lineal de muestras anteriores de la señal de entrada

$$\hat{y}(n) = -\sum_{m=1}^{M} a_m(n) y(n-m) \qquad (6.5)$$

donde $\hat{y}(n)$ representa la señal a modelar, $a_m$ representan los coeficientes AR en la etapa *m*, y *M* representa el orden del filtro AR. Los coeficientes $a_m$ óptimos deben satisfacer el principio de ortogonalidad por el cual la señal de error *e(n)* debe ser ortogonal al subespacio ocupado por *y(n − 1)*, *y(n − 2)*,....., *y(n − m)* tal que ⟨*e(n)*, *y(n − m)*⟩ = 0 para *m* = 1, 2,....., *M*.

Usando el método de autocorrelación [Akay94], se calculan los coeficientes de predicción del proceso AR mediante los retardos de autocorrelación estimados

$$\hat{r}_{yy}(k) = -\sum_{m=1}^{M} a_m \hat{r}_{yy}(k-m) \qquad k = 1,2,....,M \qquad (6.6)$$

donde *M* representa el orden del filtro AR y *k* representa la función de autocorrelación (ACF) de mayor orden. Los parámetros ACF se calculan a partir de la secuencia de entrada *x(n)* como

$$\hat{r}_{yy}(k) = \frac{1}{N} \sum_{n=0}^{N-1-k} y(n) y(n+k) \qquad k = 0,1,....,N-1 \qquad (6.7)$$

Se observa que $\hat{r}_{yy}(k) = \hat{r}_{yy}(-k)$ . La potencia de ruido (varianza) $\sigma_e^2$ se puede estimar como

$$\sigma_e^2 = \hat{r}_{yy}(0) + \sum_{m=1}^{M} a_m \hat{r}_{yy}(m) \qquad (6.8)$$

Esta ecuación es conocida como la ecuación Yule-Walker. Se puede usar la función ACF para crear una matriz de autocorrelación *R*. Los coeficientes de predicción se obtienen usando la fórmula

$$\begin{bmatrix} \hat{r}_{yy}(0) & \hat{r}_{yy}(-1) & \cdots & \hat{r}_{yy}(-M) \\ \hat{r}_{yy}(1) & \hat{r}_{yy}(0) & \cdots & \hat{r}_{yy}(1-M) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_{yy}(M) & \hat{r}_{yy}(M-1) & \cdots & \hat{r}_{yy}(0) \end{bmatrix} \begin{bmatrix} a(0) \\ a(1) \\ \vdots \\ a(M) \end{bmatrix} = \begin{bmatrix} \sigma_e^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{6.9}$$

Una vez calculados los coeficientes de predicción, se obtiene el número de bloques de *slots* que se predice demandar por cada terminal participante en la red, como una combinación lineal de las demandas anteriores según la ecuación (6.5).

Sin embargo, las peticiones de bloques de *slots* para cada terminal en un *reallocation period* son estadísticamente independientes de las peticiones de bloques de *slots* en otro *reallocation period*, con lo cual es un proceso blanco. Esto quiere decir que solo se puede predecir la media de las peticiones en los *reallocation period* pasados, es decir, el orden del filtro AR es 1. Pero pueden existir situaciones concretas en una red en las que esto no sea útil. Por ejemplo, cuando un terminal tiene una gran movilidad en la red puede entrar y salir de un escenario. En general, cuando se dan unas condiciones de variación de las necesidades de transmisión, la media cambia con el tiempo. En estos casos habría que realizar un estudio pormenorizado del orden del filtro AR, superior a 1, mas apropiado para este proceso, estudio que no es motivo de esta tesis.

## 6.3 Aplicación del Dilema del Prisionero Iterado para TSR.

Finalmente, se propone la utilización de la Teoría de Juegos mediante el Dilema del Prisionero Iterado para asignar los bloques de *slots* que quedan libres a todos los terminales participantes en la red. El objetivo, como se ha comentado anteriormente, es poder aprovechar lo máximo posible la capacidad que ofrece la red. Para ello, la idea es que cada terminal pueda pedir más bloques de *time slots* de los que realmente necesita. Sin embargo, para que el método sea óptimo, no deberían de asignarse bloques de *slots*

de más a un terminal (bloques que no vaya a utilizar) si esos bloques pueden ser necesarios para algún otro terminal de la red. Con ello, cada terminal tiene dos opciones: o pedir mas bloques de *slots* de los necesarios para que se le asignen mas bloques, o pedir solamente los bloques de *slots* que necesita. Este problema podría asemejarse al Dilema del Prisionero Iterado, expuesto en el Capítulo 4 de esta tesis, del siguiente modo:

- Cooperar: significaría no pedir más bloques de *slots* de los necesarios.

- Delatar: significaría pedir más bloques de *slots* de los necesarios para que se le asignen más bloques.

Evidentemente, para poder utilizar este planteamiento es necesario que cada terminal indique en el mensaje de petición de bloques de *slots* para el siguiente *reallocation periods* si pide más bloques de slots de los necesarios, y cuantos pide.

La asignación a un terminal de más bloques de *slots* de los necesarios sería penalizada, ya que pueden no ser utilizados por el terminal y pueden ser necesarios para algún otro terminal en la red. La penalización sería mayor (es decir, la recompensa sería menor) cuanto mayor fuese el número de bloques de *slots* asignados de más.

Para que se cumplan las restricciones del juego para dos jugadores en cuanto a recompensas a aplicar, las opciones posibles del problema son:

- Cooperar, cooperar: ningún participante en la red pide más bloques de *slots* de los necesarios.

- Delatar, delatar: los dos participantes en la red piden muchos más bloques de *slots* de los necesarios. Así la penalización es mayor (la recompensa es menor).

- Cooperar, delatar: el participante que coopera no pide más bloques de *slots* de los necesarios, y el que delata no pide muchos más bloques de los necesarios para que la penalización sea menor (la recompensa sea alta).

El método expuesto hasta ahora es para dos participantes. Pero se puede generalizar a *N* participantes [Bankes94], que es lo que realmente se necesita para nuestro problema. Para *N* participantes, las posibles opciones de cada movimiento en el juego ya no son solo cuatro, como para dos participantes (cooperar-cooperar, cooperar-delatar, delatar-cooperar, delatar-delatar). Ahora existen más combinaciones posibles para cooperar y delatar, dependiendo del número *N* de participantes. Siguiendo a Bankes, mediante un Algoritmo Genético se pretende obtener la mejor estrategia posible, es decir, la que mayor puntuación obtenga.

La decisión de un participante en cada movimiento va a depender de lo que haga el resto de participantes en el movimiento anterior. Por ello, cada estrategia se codifica para el Algoritmo Genético con *N*+1 posiciones. Las primeras *N* posiciones indican las decisiones de cooperar o delatar dependiendo del anterior movimiento: la posición *j* (*j* = 0...*N*-1) indica la decisión cuando *j* del resto de participantes cooperan en el anterior movimiento. La última posición de la estrategia indica el primer movimiento del participante en el juego.

Para generalizar el problema de 2 jugadores a *N*, las puntuaciones que se darán por las decisiones son:

- Si los *N* participantes cooperan, cada uno tiene una puntuación de 3 puntos.

- Si los *N* participantes delatan, cada uno tiene una puntuación de 1 punto.

- Si algunos participantes cooperan y otros delatan, cada uno de los que coopera tiene una puntuación de 0 puntos y cada uno de los que delata tiene una puntuación de 5 puntos.

Para obtener la mejor estrategia, se crea una población inicial de individuos (estrategias). Para obtener la puntuación final de una estrategia, ésta se enfrenta a una batería de estrategias determinada, en un juego con un número de movimientos dado. La puntuación final de esa estrategia será la suma de las puntuaciones de ese participante en cada movimiento del juego. La mejor estrategia obtenida con el Algoritmo Genético será la que mayor puntuación obtenga en el juego anterior.

# Parte III

# Interpretación y Discusión de Resultados

# Capítulo 7 Resultados obtenidos con las propuestas basadas en Técnicas de Optimización Combinatoria

## 7.1 Introducción.

En este capítulo se presentan los resultados obtenidos al realizar las distintas simulaciones para las propuestas basadas en Técnicas de Optimización Combinatoria que se proponen en el Capítulo 5 de esta tesis para la mejora del algoritmo TSR de reasignación dinámica de *slots* actual, en el caso de que la demanda total de *slots* exceda de la capacidad disponible. Con ello, se pretende eliminar los casos de conflicto en la asignación y minimizar el valor medio de la cola máxima de mensajes no transmitidos en cada terminal participante, es decir, que para un número de *reallocation periods* dado, las colas de mensajes no transmitidos en todos los participantes en una red se mantengan lo más pequeñas posibles.

Se realizan distintas simulaciones para cada método de asignación de *slots* propuesto. Para ello, se usan unos datos de entrada (peticiones de bloques de *slots* de cada participante en la red) con dos distribuciones de tráfico diferentes: una distribución de Poisson y una distribución binomial de Bernouilli.

Los resultados obtenidos se evalúan y se presentan en distintas gráficas. Estos datos corresponden al valor medio de la cola máxima de mensajes en cada participante

para todos los *reallocation periods* de la simulación. Finalmente, se realiza una discusión de estos resultados y un estudio comparativo de los distintos algoritmos propuestos con respecto al algoritmo actual de reasignación de *slots*, en referencia a las soluciones obtenidas y al grado de complejidad de cada una de ellas (dependiendo de la potencia de cálculo y la velocidad de convergencia a una solución óptima).

Para la red de Hopfield propuesta, los valores de los parámetros de la ecuación de movimiento utilizada en las simulaciones son $A = B = C = 1$, y $D = E = 5$, siguiendo [Funabiki97]. Los valores iniciales de entrada de las neuronas $U$ son enteros uniformemente aleatorios entre 0 y -100, y los valores iniciales de salida de las neuronas $V$ son 0.

El Algoritmo Genético propuesto tiene las siguientes características:

- Tamaño de cada población: 100.
- Nº generaciones: 10.
- *pcross* = 0.8.
- *pmutation* = 0.03.
- $C_{max}$ se elige como una función del número de participantes en la red $N_{nd}$, del número de bloques de *slots* disponibles en modo de acceso TSR para esos terminales $P_{nd}$, y del número de bloques de *slots* que solicita cada terminal $Rq_i$. Si la ecuación (5.1) del Capítulo 5 de esta tesis nos da el valor medio del número de bloques de *time slots* que se dejan de asignar a cada participante ($\bar{x}$), se han probado diversas combinaciones para el valor de $C_{max}$. Finalmente, el valor que ofrece mejores soluciones es $C_{max} = 10N_{nd}(\bar{x})^2$, que es el que se utiliza en las simulaciones.

Sin embargo, como ya se ha comentado en el Capítulo 5 de esta tesis, un Algoritmo Genético tiene una gran variedad de parámetros aleatorios (punto de corte en los cromosomas de los padres para crear nuevos individuos, probabilidad de cruce y mutación no es la unidad,…), que hacen que la solución no sea igual para dos simulaciones con los mismos datos de entrada. Esto quiere decir que cada terminal participante en la red puede obtener distintas asignaciones de bloques de *slots* con los mismos datos de peticiones de *slots* de todos los participantes. Por ello, se propone una

variante del Algoritmo Genético con las mismas características que el anterior, pero con todos los parámetros aleatorios fijos a un mismo valor para todas las simulaciones realizadas.

## 7.2 Simulaciones realizadas.

Las simulaciones se realizan para un número *RP* de *reallocation periods*. En un *reallocation period t*, cada terminal participante *i* se simula como un servidor con una cola de mensajes $q_i(t)$, una petición de bloques de *slots* $Rq_i(t)$ como entrada y un número de bloques de *slots* asignados $X_i(t)$ como salida (ver Fig.7.1).

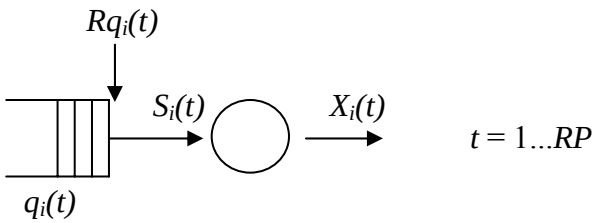



Fig.7.1: Representación del terminal *i* en el *reallocation period t*

Los números $Rq_i(t)$ de petición de bloques de *slots* para cada participante en el *reallocation period t* utilizados para las simulaciones son enteros aleatorios, usando dos distribuciones de tráfico diferentes: una distribución de Poisson y una distribución binomial de Bernouilli. El número máximo de bloques de *slots* que puede pedir un terminal en un *reallocation period* es el 60 % del número de bloques de *slots* disponibles para todos los terminales en modo de acceso TSR [MIDSCO99]. Por lo tanto, sea $y_i(t)$ el número de bloques de *slots* que no pueden ser solicitados por el terminal *i* en el *reallocation period t*. Si $P_{nd}$ es el número de bloques de *slots* disponibles para todos los terminales, las ecuaciones en este *reallocation period* son las siguientes:

- $q_i(t) = S_i(t-1) - X_i(t-1) + y_i(t-1)$
- si $Rq_i(t) + q_i(t) \leq 0.6P_{nd}$

  $S_i(t) = Rq_i(t) + q_i(t)$

  $y_i(t) = 0$
- si $Rq_i(t) + q_i(t) > 0.6P_{nd}$

  $S_i(t) = 0.6P_{nd}$

  $y_i(t) = (Rq_i(t) + q_i(t)) - 0.6P_{nd}$
- $X_i(t)$ se obtiene aplicando el algoritmo seleccionado a partir de las peticiones $S_i(t)$ de todos los participantes.

Las condiciones iniciales en $t = 0$ son:

$S_i(0) = 0$

$X_i(0) = 0$

$y_i(0) = 0$

Las simulaciones se realizan en un entorno Matlab. Para cada uno de los algoritmos propuestos se genera una matriz $RP$ x $N_{nd}$ de valores de $Rq_i$, de forma que cada fila expresa los valores de $Rq_i$ en cada *reallocation period* y cada columna expresa el valor de $Rq_i$ para cada participante en la red. Con estos valores de entrada se obtiene, para cada algoritmo propuesto, una matriz $RP$ x $N_{nd}$ con los valores $X_i(t)$ de asignación de bloques de *slots* para cada participante en la red y para cada *reallocation period*.

Para cada uno de los algoritmos propuestos, se obtiene la máxima cola de mensajes en el *reallocation period t* como $\max(q_i(t))$, $i = 1 \ldots N_{nd}$. Finalmente se calcula el valor medio de la cola máxima de mensajes en todos los *reallocation periods*.

## 7.3 Análisis de la adecuación de la red de Hopfield.

Antes que nada, se va a mostrar si la red neuronal de Hopfield propuesta puede encontrar la solución global al problema, es decir, que se asignen todos los bloques de *time slots* a los terminales participantes cuando la suma de las peticiones de bloques de

*slots* de todos los terminales participantes es mayor que el número de bloques de *slots* disponible [Vera02]. Como se comentó en el apartado 5.3 de esta tesis, la restricción impuesta de que todos los bloques de *time slots* deben ser asignados a los terminales es demasiado restrictiva para la convergencia de la red a una solución, así que se eliminó en la ecuación de movimiento.

Para ello, se examinan un total de 14 escenarios de una librería de diseños de redes Link-16, mostrados en la Tabla 7.1. Cada escenario está definido por un número de terminales no sordos participantes en la red ($N_{nd}$), un número de bloques de *slots* disponibles en modo de acceso TSR para esos terminales ($P_{nd}$) y la suma total de bloques de *slots* demandados por todos esos terminales. Para cada escenario se realizan 50 simulaciones. Como se comentó en el apartado 7.1 de este capítulo, los valores de los parámetros de la ecuación de movimiento utilizados son $A = B = C = 1$, y $D = E = 5$. Los valores iniciales de entrada de las neuronas $U$ son enteros uniformemente aleatorios entre 0 y -100, y los valores iniciales de salida de las neuronas $V$ son 0. Cada simulación se realiza para un máximo de 1000 iteraciones.

| **Ejemplo** | $N_{nd}$ | $P_{nd}$ | **Nº total de bloques de *slots* demandados** |
|---|---|---|---|
| Ejemplo 1 | 1 | 32 | 7 |
| Ejemplo 2 | 5 | 32 | 49 |
| Ejemplo 3 | 10 | 32 | 103 |
| Ejemplo 4 | 15 | 32 | 136 |
| Ejemplo 5 | 20 | 32 | 206 |
| Ejemplo 6 | 25 | 32 | 228 |
| Ejemplo 7 | 30 | 32 | 278 |
| Ejemplo 8 | 5 | 4 | 10 |
| Ejemplo 9 | 5 | 8 | 12 |
| Ejemplo 10 | 5 | 16 | 20 |
| Ejemplo 11 | 5 | 24 | 35 |
| Ejemplo 12 | 5 | 32 | 40 |
| Ejemplo 13 | 5 | 48 | 64 |
| Ejemplo 14 | 5 | 64 | 100 |

Tabla 7.1: Especificaciones de los ejemplos simulados

La Tabla 7.2 resume los resultados de las simulaciones realizadas. Las peticiones de bloques de *slots* de los terminales participante en la red son enteros aleatorios con una distribución de tráfico de Poisson. Las simulaciones se realizan para un *reallocation period*.

Esta tabla muestra, para cada ejemplo, el número máximo, mínimo y el valor medio del número de iteraciones que la red de Hopfield necesita para la convergencia, así como el número máximo, mínimo y el valor medio del número de bloques de *slots* asignados en todas las simulaciones realizadas. En todos los ejemplos se observa que la red de Hopfield puede encontrar la solución global, es decir, puede asignar todos los bloques de *slots* disponibles a los teminales, y converge a esta solución dentro de un margen de 200 iteraciones.

| **Ejemplo** | **Nº iteraciones para convergencia** | | | **Nº total de bloques de *slots* asignados** | | |
|---|---|---|---|---|---|---|
| | **Máximo** | **Mínimo** | **V. Medio** | **Maximum** | **Minimum** | **V. Medio** |
| Ejemplo 1 | 1 | 1 | 1 | 7 | 7 | 7 |
| Ejemplo 2 | 95 | 15 | 56.5 | 32 | 31 | 31.8 |
| Ejemplo 3 | 102 | 17 | 47.6 | 32 | 32 | 32 |
| Ejemplo 4 | 174 | 6 | 48.1 | 32 | 31 | 31.9 |
| Ejemplo 5 | 141 | 6 | 45.3 | 32 | 31 | 31.9 |
| Ejemplo 6 | 187 | 6 | 56.8 | 32 | 31 | 31.8 |
| Ejemplo 7 | 202 | 10 | 55.8 | 32 | 31 | 31.8 |
| Ejemplo 8 | 87 | 6 | 27.4 | 4 | 3 | 3.9 |
| Ejemplo 9 | 154 | 6 | 24.5 | 8 | 7 | 7.9 |
| Ejemplo 10 | 139 | 9 | 33 | 16 | 15 | 15.9 |
| Ejemplo 11 | 178 | 10 | 57.9 | 24 | 23 | 23.8 |
| Ejemplo 12 | 168 | 13 | 55.1 | 32 | 31 | 31.8 |
| Ejemplo 13 | 197 | 14 | 62 | 48 | 47 | 47.8 |
| Ejemplo 14 | 201 | 22 | 75.1 | 64 | 63 | 63.8 |

Tabla 7.2: Resultados de las simulaciones

## 7.4 Resultados de las simulaciones y discusión.

Se realizan varias simulaciones para cada algoritmo propuesto, variando los distintos parámetros de la red: el número de terminales no sordos participantes en la red ($N_{nd}$), el número de bloques de *slots* disponibles en modo de acceso TSR para esos terminales ($P_{nd}$) y el número de *reallocation periods* (*RP*) [Vera02]. Los resultados del valor medio de la cola máxima de mensajes en función de estos parámetros se representan en las Figs. 7.2, 7.3, 7.4 (con una distribución de tráfico de entrada de Poisson) y en las Figs. 7.5, 7.6, 7.7 (con una distribución de tráfico de entrada binomial de Bernouilli). Se representan los resultados de las simulaciones para el algoritmo TSR actual, el método de Branch and Bound, la red neuronal de Hopfield, el algoritmo heurístico y el Algoritmo Genético. En el caso del Algoritmo Genético, se han realizado simulaciones para el algoritmo genérico y la variante del algoritmo con todos los parámetros aleatorios fijos a un mismo valor. Los resultados de las simulaciones son similares, por lo que en las figuras solo se representan los resultados obtenidos con la variante del algoritmo.

Para cada algoritmo propuesto y cada distribución de tráfico de entrada, se calcula la media de los resultados obtenidos y el intervalo de confianza del 95%. La Fig. 7.2 muestra los resultados de las simulaciones con el valor de la media y el intervalo de confianza. Sin embargo, este intervalo es similar en todas las simulaciones, por lo que las siguientes figuras solo muestran el valor medio para cada algoritmo.

Además de estos resultados, también se obtiene la carga computacional que cada algoritmo propuesto necesita para encontrar la solución al problema. La Fig. 7.8 muestra el tiempo (en segundos) que tarda cada algoritmo en converger a la solución en las simulaciones realizadas, variando el número de terminales no sordos participantes en la red $N_{nd}$ con una distribución de tráfico de entrada de Poisson. El objetivo es realizar un estudio comparativo entre los distintos algoritmos en referencia a la velocidad de convergencia a una solución óptima. El algoritmo TSR actual y el algoritmo heurístico son muy rápidos, por lo que los valores obtenidos están próximos a cero. Los resultados de las demás simulaciones son similares.

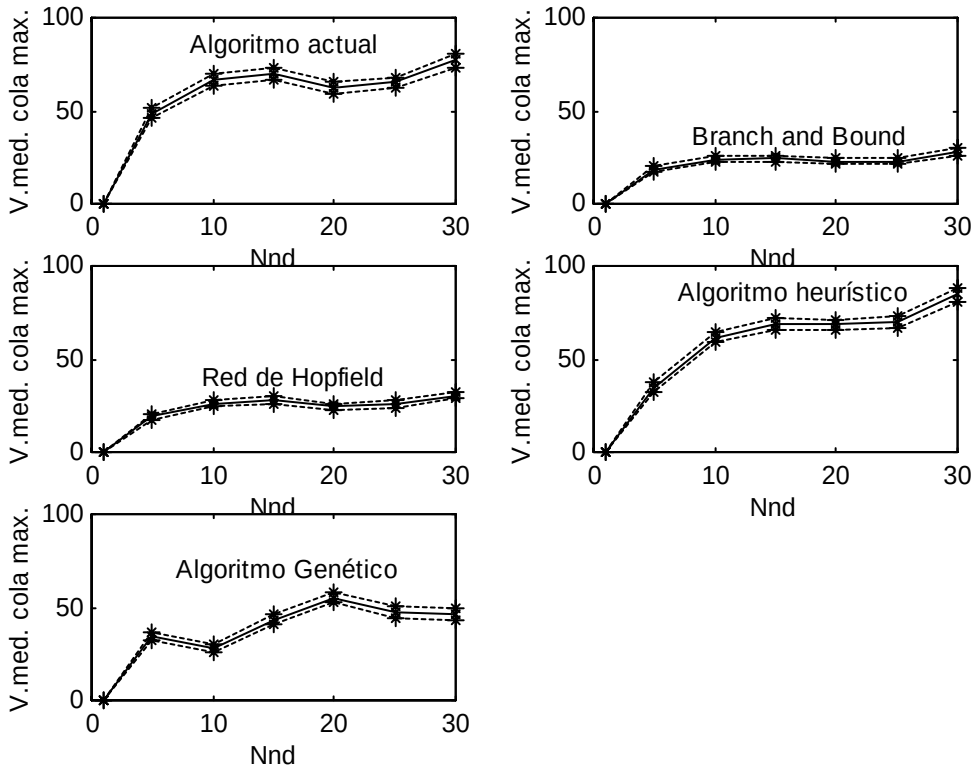



Fig.7.2: Comparación entre todos los algoritmos variando $N_{nd}$, con tráfico de Poisson

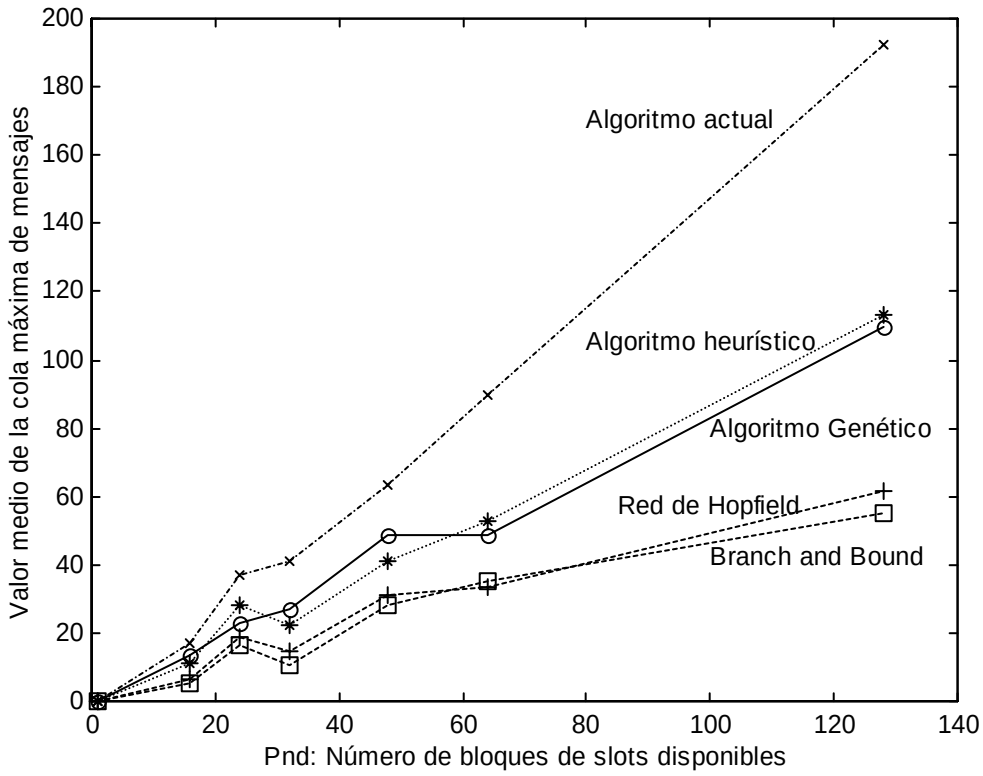



Fig.7.3: Comparación entre todos los algoritmos variando $P_{nd}$, con tráfico de Poisson

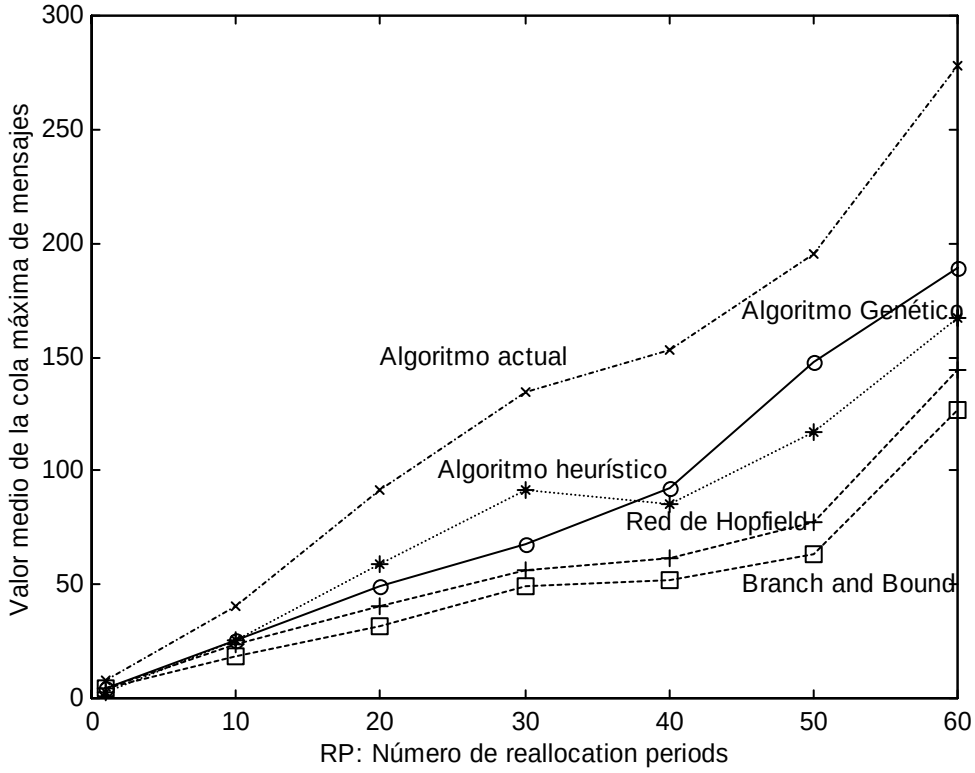



Fig.7.4: Comparación entre todos los algoritmos variando *RP*, con tráfico de Poisson

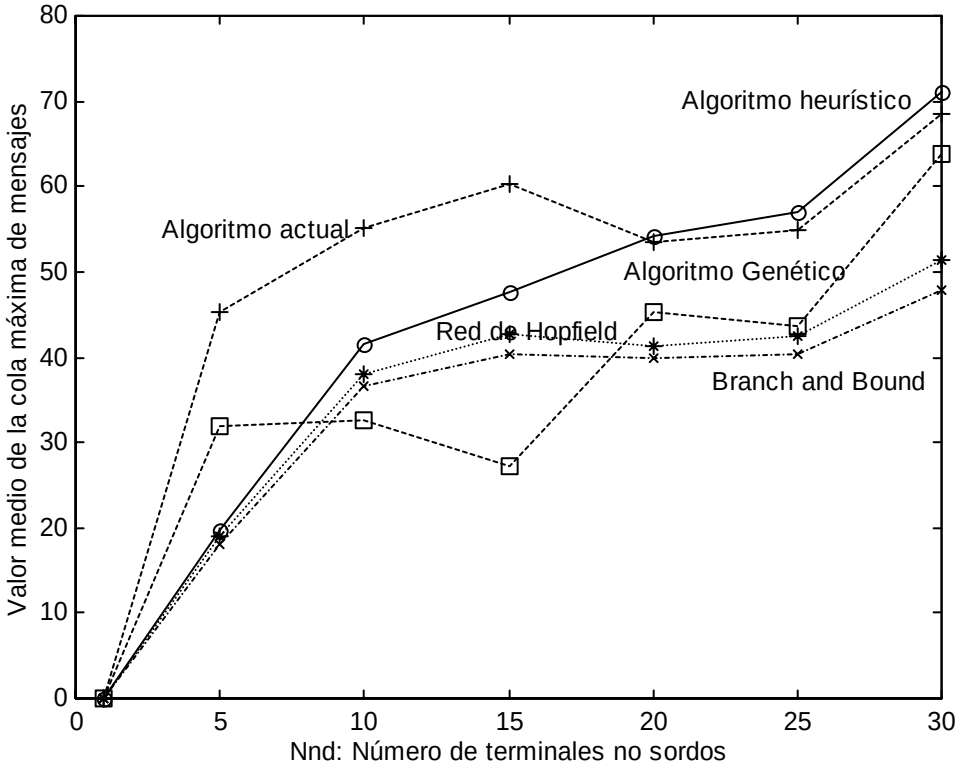



Fig.7.5: Comparación entre todos los algoritmos variando $N_{nd}$, con tráfico de Bernouilli

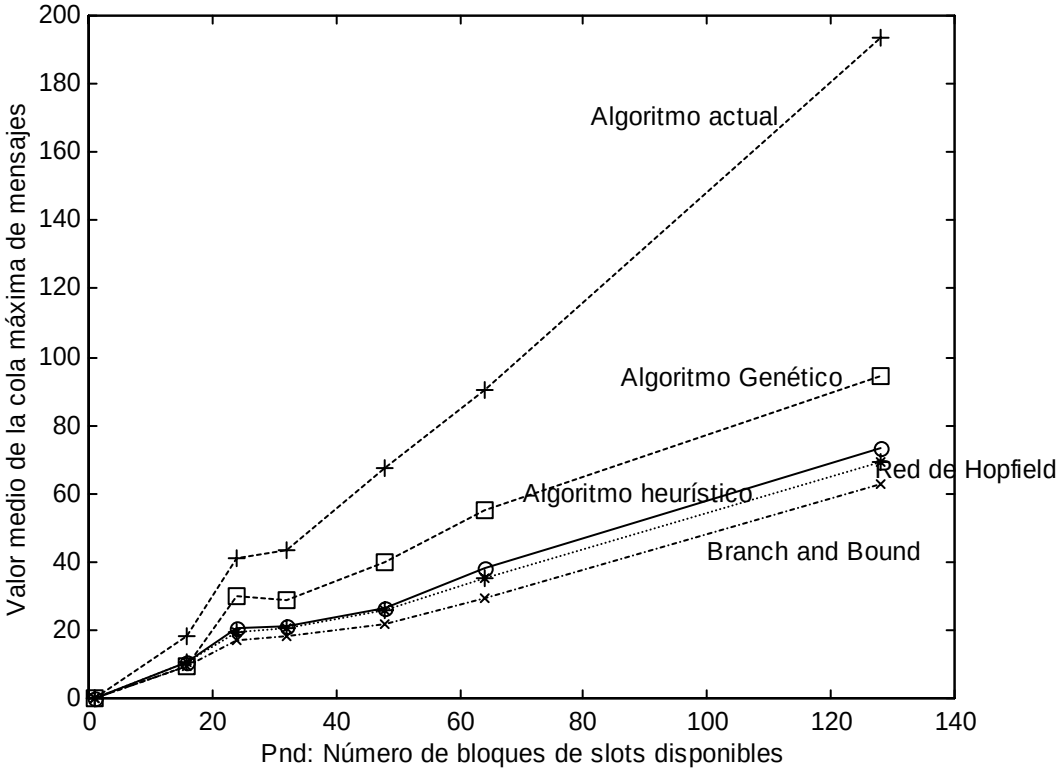



Fig.7.6: Comparación entre todos los algoritmos variando $P_{nd}$, con tráfico de Bernouilli

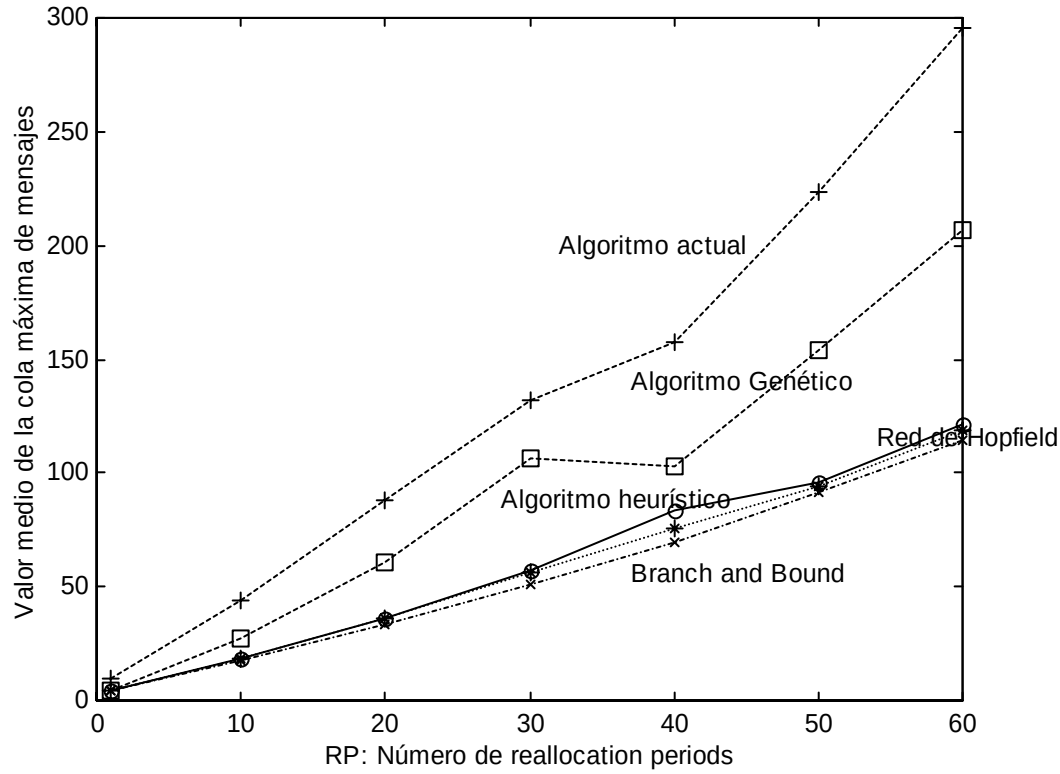



Fig.7.7: Comparación entre todos los algoritmos variando *RP*, con tráfico de Bernouilli

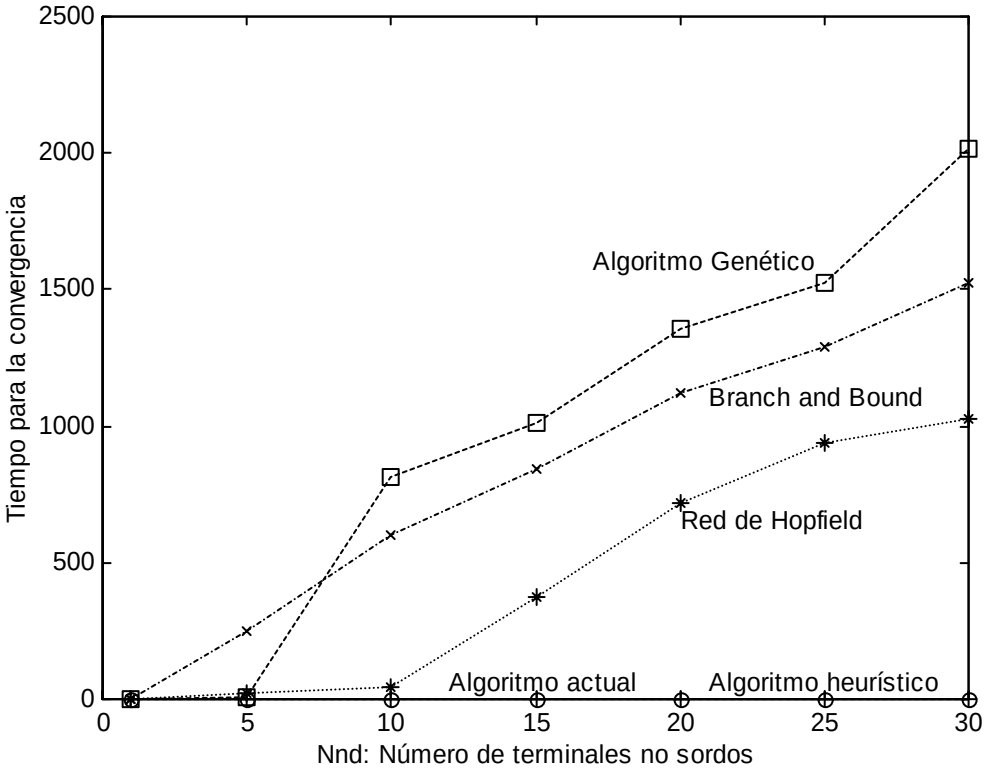



Fig.7.8: Comparación entre todos los algoritmos variando $N_{nd}$, con tráfico de Poisson

Las Figs. 7.2 y 7.5 muestran los resultados obtenidos en las simulaciones variando el número de terminales no sordos participantes en la red $N_{nd}$. El número de bloques de *slots* disponibles para TSR es $P_{nd}$ = 32 y el número de *reallocation periods* es $RP$ = 10. El valor medio de la cola máxima de mensajes usando el método Branch and Bound, la red neuronal de Hopfield y el Algoritmo Genético, es siempre menor que el valor medio de la cola máxima de mensajes usando el algoritmo TSR actual. El valor medio de la cola máxima de mensajes usando el algoritmo heurístico es menor que el valor medio de la cola máxima de mensajes usando el algoritmo TSR actual cuando el número de participantes no sordos $N_{nd}$ es menor de 15. La diferencia entre los resultados de las simulaciones para los dos algoritmos disminuye al aumentar el valor de $N_{nd}$. Esto es debido a que, en el algoritmo TSR actual, una vez asignados todos los bloques de *slots* disponibles, a los terminales que faltan por asignar se les asigna un bloque de *slots*, permitiendo conflictos. También se permiten conflictos si la suma de las peticiones de los terminales sordos es mayor que el número de bloques de *slots* disponibles para TSR. En el algoritmo heurístico se eliminan los casos de conflictividad asignando cero

bloques de *slots* a los terminales que faltan por asignar. Si el número de terminales sordos es cero, o la suma de peticiones de todos los terminales sordos es menor o igual que el número de bloques de *slots* disponibles para TSR, el resultado de las simulaciones usando el algoritmo heurístico es menor que el resultado usando el algoritmo TSR actual. Finalmente, el resultado de las simulaciones usando la red neuronal de Hopfield es muy similar al resultado de las simulaciones usando el método Branch and Bound. El resultado de las simulaciones usando el Algoritmo Genético también es menor que el resultado de las simulaciones usando el algoritmo TSR actual.

Las Figs. 7.3 y 7.6 muestran los resultados obtenidos en las simulaciones variando el valor del número de bloques de *slots* disponibles para TSR $P_{nd}$. El número de terminales no sordos participantes en la red es $N_{nd} = 5$ y el número de *reallocation periods* es $RP = 10$. Los resultados de las simulaciones realizadas para cualquier otro valor de $N_{nd} < 15$ son similares. Se observa que el mejor algoritmo es el método Branch and Bound. El valor medio de la cola máxima de mensajes usando el método Branch and Bound y la red neuronal de Hopfield es siempre menor que el valor medio de la cola máxima de mensajes usando el resto de algoritmos. Todos ellos mejoran el algoritmo TSR actual, ya que el resultado de las simulaciones usando cualquier método es menor que el resultado usando el algoritmo TSR actual. La diferencia entre los resultados de las simulaciones para todos los algoritmos aumenta a medida que aumenta el valor de $P_{nd}$.

Las Figs. 7.4 y 7.7 muestran los resultados obtenidos en las simulaciones variando el valor del número de *reallocation periods RP*. El número de terminales no sordos participantes en la red es $N_{nd} = 5$ y el número de bloques de *slots* disponibles para TSR es $P_{nd} = 32$. Los resultados de las simulaciones realizadas para cualquier otro valor de $N_{nd} < 15$ son similares. De nuevo se observa que el mejor algoritmo es el método Branch and Bound. El valor medio de la cola máxima de mensajes usando el método Branch and Bound y la red neuronal de Hopfield es siempre menor que el valor medio de la cola máxima de mensajes usando el resto de algoritmos. Todos ellos mejoran el algoritmo TSR actual, ya que el resultado de las simulaciones usando cualquier método es menor que el resultado usando el algoritmo TSR actual. La diferencia entre los resultados de las simulaciones para todos los algoritmos aumenta a medida que aumenta el valor de *RP*.

La Fig. 7.8 y las simulaciones realizadas muestran que el método Branch and Bound y el Algoritmo Genético son muy complejos, lentos y necesitan una alta carga computacional para resolver el problema. En el caso de la red neuronal de Hopfield, para valores altos de los parámetros ($N_{nd} > 10$ y $P_{nd} > 48$, es decir, una red neuronal de más de 480 (10 x 48) neuronas), el tiempo necesario para la convergencia a la solución es muy alto. Finalmente, el algoritmo heurístico es el más rápido, junto con el algoritmo TSR actual.

En resumen, los resultados de las simulaciones muestran que todos los métodos propuestos en esta tesis para el caso de que la demanda total de *slots* exceda de la capacidad disponible mejoran el algoritmo TSR actual, excepto el algoritmo heurístico para un número de participantes mayor de 15. El método de Branch and Bound y la red neuronal de Hopfield son los que mejores resultados obtienen, y el algoritmo heurístico el que peor resultado obtiene. Sin embargo, la ventaja de este último algoritmo es la sencillez, rapidez y menor cálculo frente al resto de propuestas, que son más lentas y necesitan una alta carga computacional. Además, estos resultados son independientes del tipo de distribución del tráfico de datos de entrada.

# Capítulo 8 Resultados obtenidos con las propuestas basadas en Técnicas Predictivas y Teoría de Juegos

## 8.1 Introducción.

En este capítulo se presentan los resultados obtenidos al realizar las distintas simulaciones para las propuestas basadas en Técnicas Predictivas y Teoría de Juegos que se proponen en el Capítulo 6 de esta tesis para la mejora del algoritmo TSR de reasignación dinámica de *slots* actual, en el caso de que la demanda total de *slots* sea menor que la capacidad disponible. Con ello, se pretende predecir la demanda de *slots* de cada terminal en función de las necesidades en los últimos *reallocation periods*, asignando al terminal parte de los bloques de *slots* que quedan libres en la red, además de los demandados por él. Esto hace que se minimice el tiempo de transmisión de los mensajes del terminal desde que se demandan los *slots* hasta que se transmiten, ya que, al asignarle más bloques de *slots*, puede transmitir antes en el siguiente *reallocation period*.

Se realizan distintas simulaciones para cada método de asignación de *slots* propuesto. Para ello, se usan unos datos de entrada (peticiones de bloques de *slots* de cada participante en la red) con dos distribuciones de tráfico diferentes: una distribución de Poisson y una distribución binomial de Bernouilli. Todas las simulaciones se realizan en un entorno Matlab.

Los resultados obtenidos se evalúan y se presentan en distintas gráficas. Estos datos corresponden al valor medio del tiempo que transcurre entre el momento en que se solicitan los *slots* en un *reallocation period* y se transmiten los mensajes en los siguientes *reallocation periods*. Finalmente, se realiza una discusión de estos resultados y un estudio comparativo de los distintos algoritmos propuestos con respecto al algoritmo actual de reasignación de *slots*, en referencia a las soluciones obtenidas.

Para el método autoregresivo (AR) propuesto, como ya se comentó en el apartado 6.2 de esta tesis, se obtiene el número de bloques de *slots* que se predice demandar por parte de los terminales en cada *reallocation period* como el valor medio del número de bloques de *slots* demandados por los terminales en los últimos *reallocation periods*, es decir, el orden del filtro AR es 1. El número de muestras que se utiliza es 5, es decir, se utiliza el número de bloques de *slots* demandados por los terminales en los 5 últimos *reallocation periods*.

Para el Dilema del Prisionero Iterado para $N$ jugadores, las simulaciones realizadas para obtener la mejor estrategia tienen las siguientes características:

- Tamaño de cada población en el Algoritmo Genético: 100.
- Nº generaciones en el Algoritmo Genético: 50.
- Cada juego comprende 150 movimientos.
- Para cada valor de $N$, se repite 40 veces el Algoritmo Genético para eliminar la variabilidad de los resultados, y se elige como estrategia ganadora la que más veces haya ganado.

## 8.2 Estrategia ganadora de la Teoría de Juegos.

Como se comentó en el apartado 6.3 de esta tesis, para obtener la mejor estrategia, se crea una población inicial de individuos (estrategias). Para obtener la puntuación final de una estrategia, ésta se enfrenta a una batería de estrategias obtenida de forma aleatoria, en un juego con un número de movimientos dado. La puntuación final de esa estrategia será la suma de las puntuaciones de ese participante en cada

movimiento del juego. La mejor estrategia obtenida con el Algoritmo Genético será la que mayor puntuación obtenga en el juego anterior.

Las simulaciones se han realizado variando el número de participantes *N*, con las características expuestas en el apartado anterior de este capítulo. Para un valor dado de *N* se obtiene una estrategia ganadora. De los resultados de estas simulaciones se obtienen las siguientes conclusiones:

- Se debe cooperar siempre en el primer movimiento.
- Para un valor de $N = 2$ se obtiene la solución del dilema del prisionero iterado para dos jugadores, es decir, hacer siempre lo que haga el otro participante en el movimiento anterior.
- Para valores pequeños de *N* (un máximo de 10 participantes), hacer siempre lo que haga la mayoría de los demás participantes en el movimiento anterior.
- Para mas de 10 participantes ($N > 10$), cooperar solo si han cooperado todos los demás participantes en el anterior movimiento. En los demás casos, delatar.

## 8.3 Resultados de las simulaciones y discusión.

Se realizan varias simulaciones para cada método de asignación de bloques de *slots* propuesto, variando los distintos parámetros de la red: el número de terminales no sordos participantes en la red ($N_{nd}$), el número de bloques de *slots* disponibles en modo de acceso TSR para esos terminales ($P_{nd}$) y el número de *reallocation periods* (*RP*). Los resultados del valor medio del tiempo que transcurre entre el momento en que se solicitan los *slots* en un *reallocation period* y se transmiten los mensajes en los siguientes *reallocation periods*, en función de estos parámetros, se representan en las Figs. 8.1, 8.2, 8.3 (con una distribución de tráfico de entrada de Poisson) y en las Figs. 8.4, 8.5, 8.6 (con una distribución de tráfico de entrada binomial de Bernouilli). Se representan los resultados de las simulaciones para el algoritmo TSR actual, el método de la estimación de la media y la estrategia ganadora en el Dilema del Prisionero Iterado para *N* jugadores.

En el caso de las Técnicas Predictivas, además del método de la estimación de la media, en el que se asigna a cada terminal participante el valor medio del número de bloques de *slots* demandados por ese terminal en los 5 últimos *reallocation periods*, se han realizado simulaciones asignando los bloques de *slots* de forma proporcional al valor medio del número de bloques de *slots* demandados por cada terminal en los 5 últimos *reallocation periods*, dependiendo del instante de transmisión del mensaje que cada terminal transmite indicando su demanda de *slots* para el siguiente *reallocation period* (hay que recordar que si se transmite el mensaje al comienzo del *reallocation period*, las probabilidades de conocer la demanda de *slots* para el siguiente *reallocation period* de forma correcta serán más pequeñas que si se transmite el mensaje al final del *reallocation period*). Los resultados de las simulaciones para estos dos métodos son similares, por lo que en las figuras sólo se representan los resultados obtenidos con el primer método.

En cuanto a la carga computacional que cada método propuesto necesita para encontrar la solución al problema, los dos métodos son bastante rápidos, como el algoritmo TSR actual. El tiempo que tarda cada método en converger a la solución, para todas las simulaciones realizadas, está próximo a cero.

Las Figs. 8.1 y 8.4 muestran los resultados obtenidos en las simulaciones variando el número de terminales no sordos participantes en la red $N_{nd}$. El número de bloques de *slots* disponibles para TSR es $P_{nd}$ = 128 y el número de *reallocation periods* es $RP$ = 10. Las Figs. 8.2 y 8.5 muestran los resultados obtenidos en las simulaciones variando el valor del número de bloques de *slots* disponibles para TSR $P_{nd}$. El número de terminales no sordos participantes en la red es $N_{nd}$ = 5 y el número de *reallocation periods* es $RP$ = 10. Finalmente, las Figs. 8.3 y 8.6 muestran los resultados obtenidos en las simulaciones variando el valor del número de *reallocation periods RP*. El número de terminales no sordos participantes en la red es $N_{nd}$ = 5 y el número de bloques de *slots* disponibles para TSR es $P_{nd}$ = 128. El valor medio del tiempo que transcurre entre el momento en que se solicitan los *slots* en un *reallocation period* y se transmiten los mensajes en los siguientes *reallocation periods* es siempre menor en los dos métodos de asignación propuestos que en el algoritmo TSR actual. Entre los dos métodos propuestos, el que mejores resultados obtiene es el método AR de estimación de la media, con unos resultados menores que el Dilema del Prisionero Iterado.

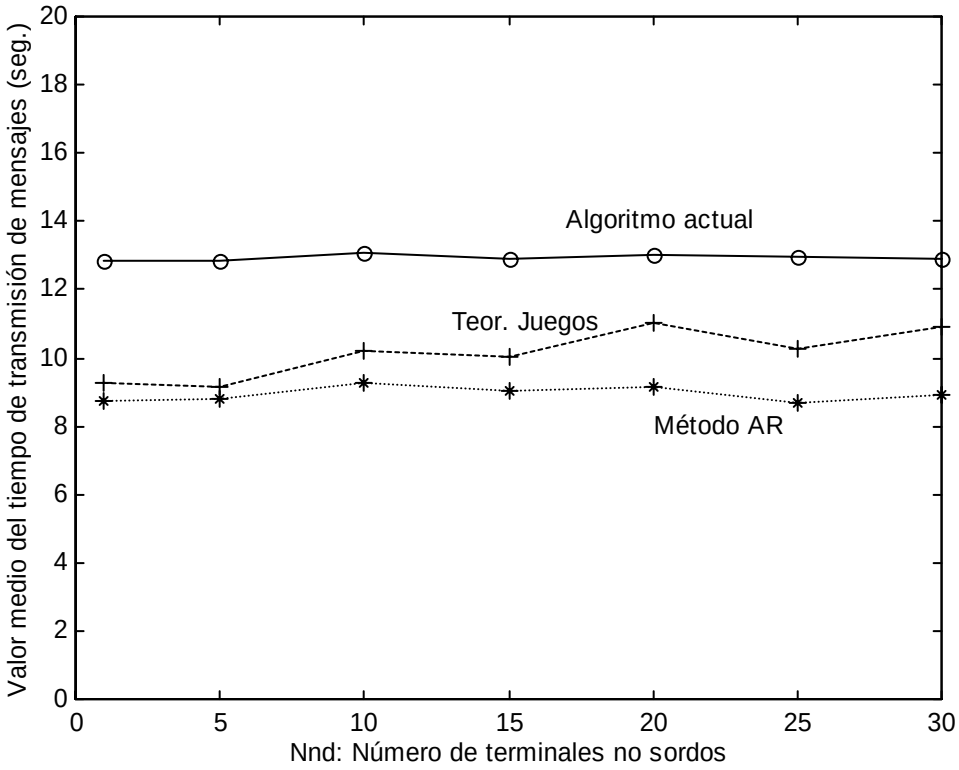



Fig.8.1: Comparación entre todos los métodos variando $N_{nd}$, con tráfico de Poisson

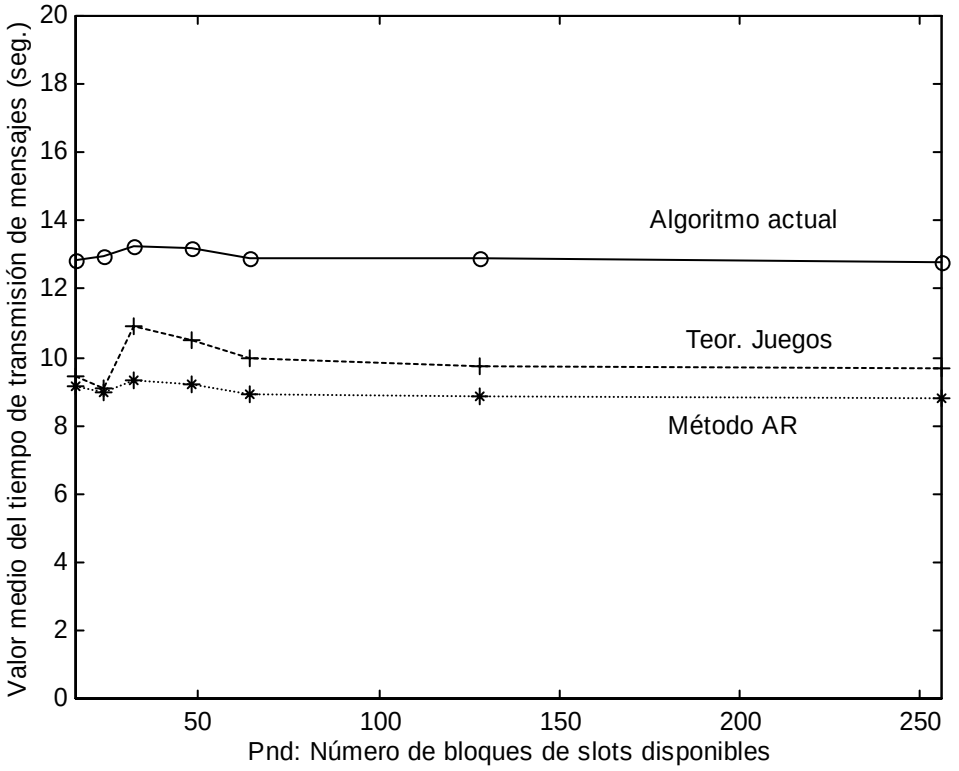



Fig.8.2: Comparación entre todos los métodos variando $P_{nd}$, con tráfico de Poisson

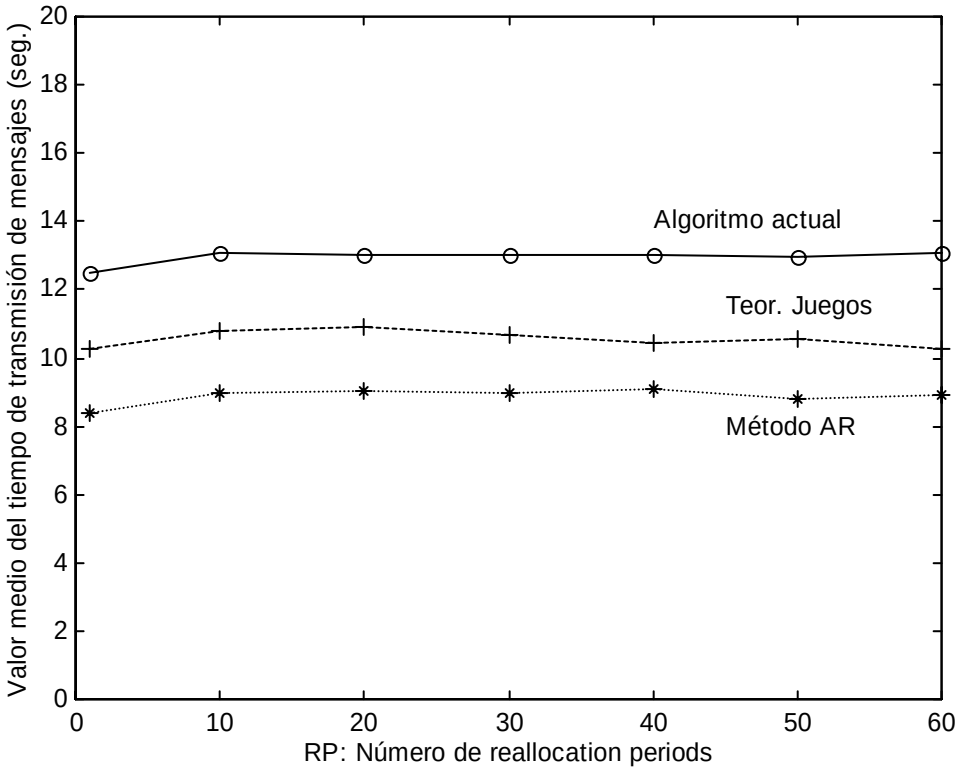



Fig.8.3: Comparación entre todos los métodos variando *RP*, con tráfico de Poisson

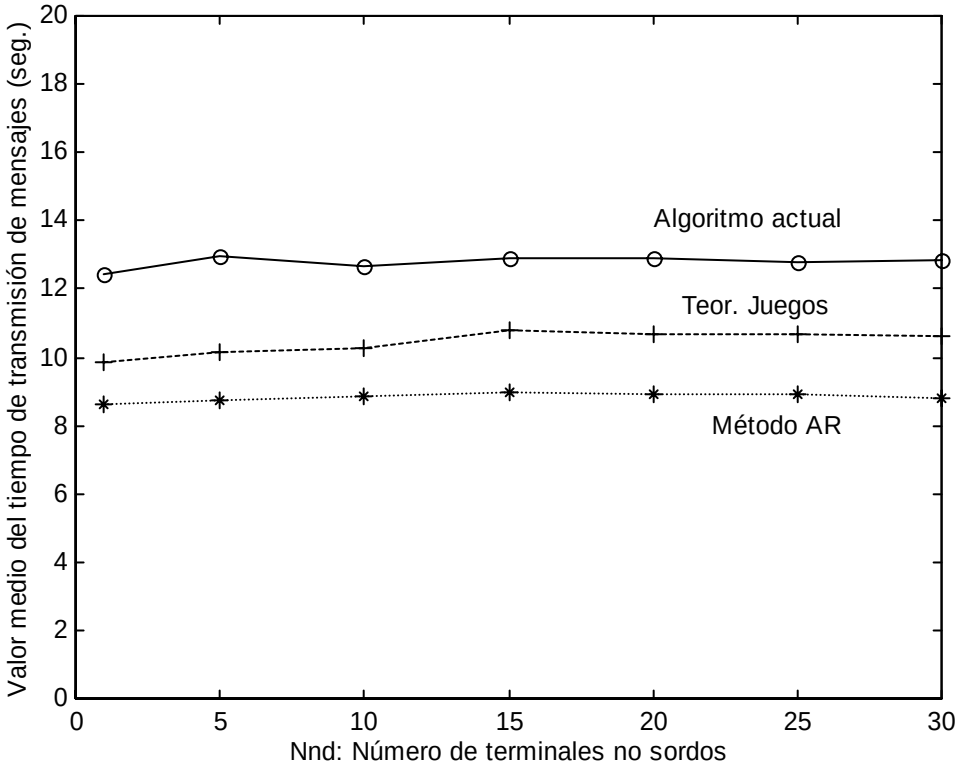



Fig.8.4: Comparación entre todos los métodos variando $N_{nd}$, con tráfico de Bernouilli

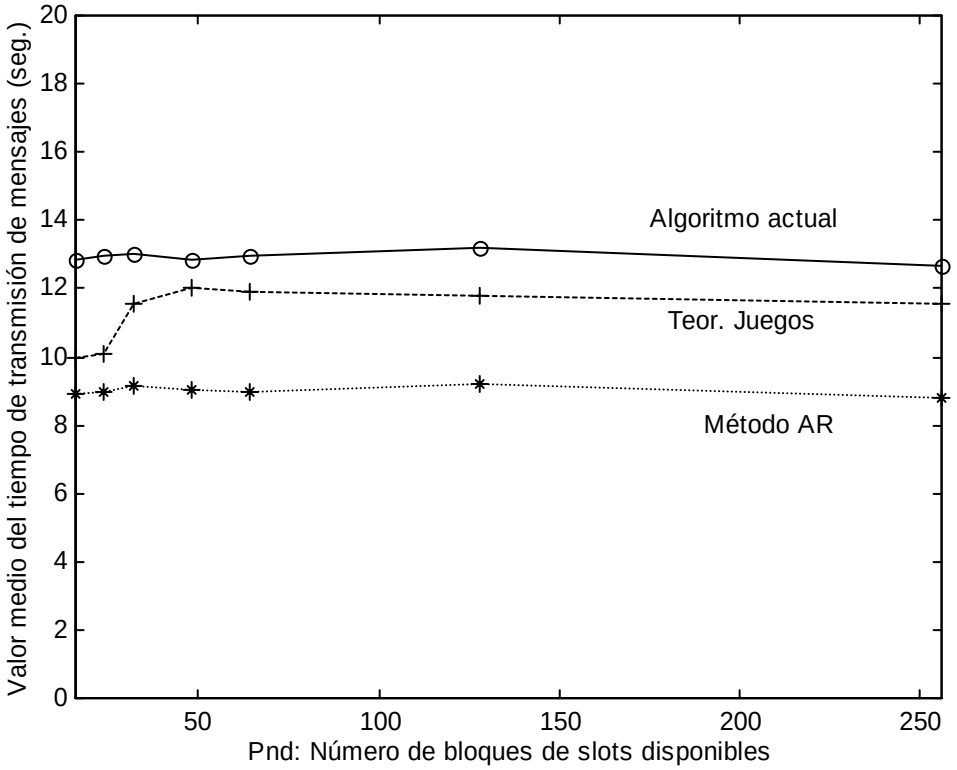



Fig.8.5: Comparación entre todos los métodos variando $P_{nd}$, con tráfico de Bernouilli

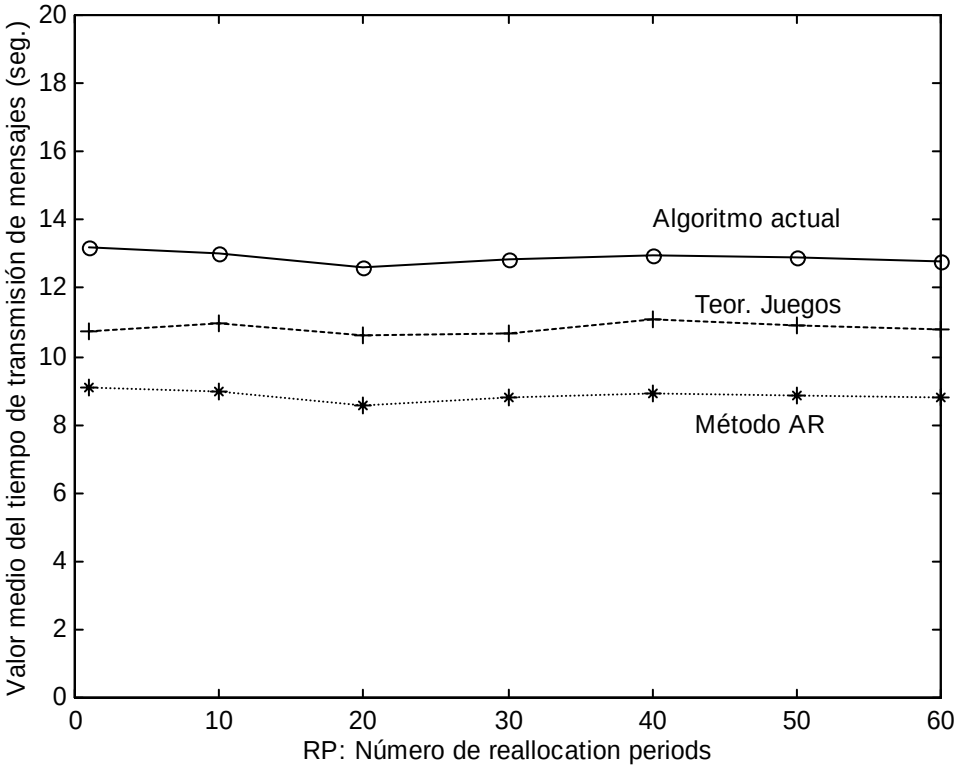



Fig.8.6: Comparación entre todos los métodos variando *RP*, con tráfico de Bernouilli

Para realizar las simulaciones se ha tomado una duración para los *reallocation periods* de 12 segundos, que corresponde a un *frame*, unidad de diseño de una red Link-16. Los resultados de estas simulaciones muestran que, en el caso del algoritmo TSR actual, el valor medio del tiempo que transcurre entre el momento en que se solicitan los *slots* demandados y se transmiten los mensajes es algo mayor de 12 segundos en todos los casos. Esto quiere decir que se tarda algo más de un *frame* de media en transmitir los mensajes. Sin embargo, para el caso de los dos métodos propuestos, los resultados de estas simulaciones muestran que el valor medio del tiempo que transcurre entre el momento en que se solicitan los *slots* demandados y se transmiten los mensajes es menor de 12 segundos en todos los casos. Esto quiere decir que se tarda menos de un *frame* de media en transmitir los mensajes, con lo que se transmiten antes que con el algoritmo TSR actual.

En resumen, los resultados de las simulaciones muestran que todos los métodos propuestos en esta tesis mejoran el algoritmo TSR actual, en el caso de que la demanda total de *slots* sea menor que la capacidad disponible. El método de estimación de la media es el que mejores resultados obtiene en todos los casos. Además, todos los métodos son bastante rápidos a la hora de converger a la solución, como el algoritmo TSR actual. Estos resultados son independientes del tipo de distribución del tráfico de datos de entrada.

Para finalizar el estudio de los algoritmos propuestos en la situación de demanda total de *slots* menor que la capacidad disponible, se han realizado distintas simulaciones incrementando la demanda de *slots* de los terminales participantes en la red, desde una demanda para un régimen de carga de la red medio, hasta llegar al límite de máxima carga de la red en el que la demanda total de *slots* sea igual a la capacidad disponible, y se observa el comportamiento del tiempo de transmisión de los mensajes de los terminales desde que se demandan los *slots* hasta que se transmiten para todos los algoritmos.

Se realizan las simulaciones para los mismos casos vistos anteriormente, es decir, con los mismos valores del número de terminales no sordos participantes en la red $N_{nd}$, número de bloques de *slots* disponibles para TSR $P_{nd}$ y número de *reallocation periods RP*, pero variando la demanda de *slots* de los terminales, desde una demanda

pequeña hasta superar la capacidad disponible de la red. El comportamiento del tiempo de transmisión de los mensajes desde que se demandan los *slots* hasta que se transmiten para todos los algoritmos es el mismo en todos los casos, por lo que solo se muestra, en la Fig. 8.7, el resultado de la simulación en la que el número de terminales no sordos participantes en la red es $N_{nd}$ = 10, el número de bloques de *slots* disponibles para TSR es $P_{nd}$ = 128, el número de *reallocation periods* es *RP* = 10 y la distribución de tráfico de entrada es de Poisson.

En dicha figura, se representa el valor medio del tiempo que transcurre entre el momento en que se solicitan los *slots* en un *reallocation period* y se transmiten los mensajes en los siguientes *reallocation periods* en función de la demanda total de *slots* en la red.

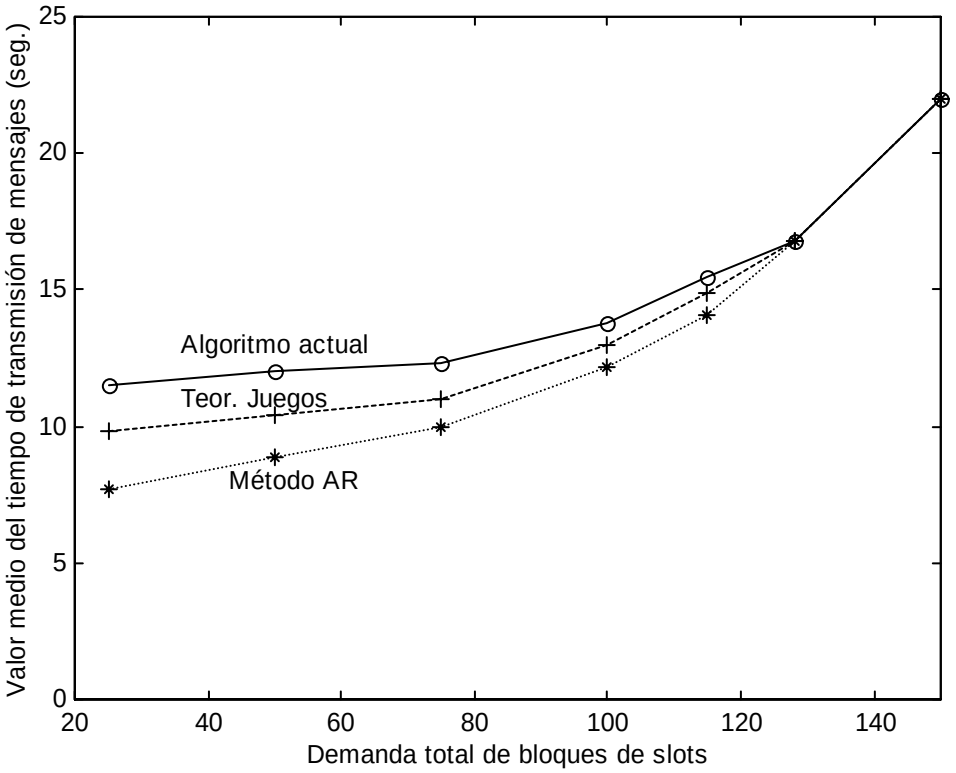



Fig.8.7: Comparación entre todos los métodos variando la demanda total de *slots*

Como puede observarse en la figura, el tiempo necesario para transmitir los mensajes aumenta para todos los algoritmos a medida que aumenta la demanda de *slots*, siendo siempre menor en los dos métodos de asignación propuestos que en el algoritmo TSR actual. Sin embargo, el resultado obtenido para los dos métodos propuestos se va aproximando cada vez más al resultado obtenido para el algoritmo TSR actual, de forma que, cuando la demanda total de *slots* es igual a la capacidad disponible de la red ($P_{nd}$), los resultados para todos los algoritmos son iguales. Esto es debido a que, en esta situación, los dos métodos de asignación propuestos no pueden asignar a los terminales más bloques de *slots* que los demandados, ya que no existen bloques de *slots* libres, con lo que la asignación es la misma que para el algoritmo TSR actual, es decir, a cada terminal se le asigna el número de bloques de *slots* demandado. Cuando la demanda total de *slots* supera a la capacidad disponible de la red, el tiempo necesario para la transmisión de los mensajes aumenta bastante porque ya empiezan a aparecer las colas de mensajes que no pueden transmitirse en el siguiente *reallocation period*. Se pasa a la situación tratada en los Capítulos 5 y 7 de esta tesis en la que la demanda total de *slots* excede de la capacidad disponible.

# Parte IV

# Conclusiones y Líneas Futuras de Trabajo

# Capítulo 9 Conclusiones y Líneas Futuras de Trabajo

## 9.1 Conclusiones y aportaciones.

Las redes de comunicaciones de datos tácticas militares tienen como objetivo fundamental el intercambio de información entre un gran número de unidades tácticas dispersas en un área, optimizando las funciones militares operativas conjuntas. Link-16 es actualmente el estándar más avanzado de este tipo de redes de la OTAN, y se está empezando a implantar en las primeras plataformas españolas. Concretamente, en las nuevas fragatas F-100 y los buques LPD de la Armada, y los aviones de combate EF-2000 y los Sistemas de Mando y Control del Ejército del Aire.

Link-16 utiliza una arquitectura de Acceso Múltiple por División en el Tiempo (TDMA), en donde a cada participante en la red se le asigna un grupo de intervalos de tiempo (*time slots*) para transmitir un determinado tipo de mensajes. Actualmente, los terminales tienen implementados dos modos de acceso a esos *time slots*: el acceso dedicado y el acceso por contienda. Para finales de este año se tiene previsto el uso del tercer modo de acceso: el acceso por reasignación dinámica de *time slots* (*Time Slot Reallocation*). Eso conlleva la implementación del algoritmo de reasignación dinámica de *time slots* en todos los terminales.

En esta tesis se estudia la mejora del algoritmo TSR (*Time Slot Reallocation*) de reasignación dinámica de *time slots* para una red de comunicaciones táctica militar Link-16, en las dos situaciones posibles: que la suma de las demandas de bloques de

*slots* de todos los terminales participantes en la red sea mayor que el número de bloques de *slots* disponible, y que la suma de las demandas de bloques de *slots* de todos los terminales sea menor que el número de bloques de *slots* disponible.

En el caso de que la demanda total de bloques de *slots* para todos los terminales sea mayor que el número de bloques de *slots* disponible, se propone la utilización de diversas técnicas de Optimización Combinatoria. Con el uso de estas técnicas, se pretende eliminar los casos de conflicto en la asignación y minimizar el valor medio de la cola máxima de mensajes no transmitidos en cada terminal participante, es decir, que para un número de *reallocation periods* dado, las colas de mensajes no transmitidos en todos los participantes en una red se mantengan lo más pequeñas posibles.

Para resolver el problema, se han planteado diversas propuestas: una propuesta basada en el método de optimización de programación entera Branch and Bound, una propuesta basada en Redes de Hopfield, un algoritmo heurístico y un Algoritmo Genético, y se ha realizado un estudio comparativo entre ellas en referencia a las soluciones obtenidas y al grado de complejidad (dependiendo de la potencia de cálculo y la velocidad de convergencia a una solución óptima).

Los resultados de las simulaciones muestran que todos los métodos propuestos mejoran el algoritmo TSR actual, es decir, reducen las colas máximas de transmisión, excepto el algoritmo heurístico para un número de participantes mayor de 15. El método de Branch and Bound y la red neuronal de Hopfield son los que mejores resultados obtienen, y el algoritmo heurístico el que peor resultado obtiene. Sin embargo, la ventaja de este último algoritmo es la sencillez, rapidez y menor cálculo frente al resto de propuestas, que son más lentas y necesitan una alta carga computacional. Además, estos resultados son independientes del tipo de distribución del tráfico de datos de entrada.

En el caso de que la demanda total de bloques de *slots* para todos los terminales sea menor que la capacidad disponible, se propone la utilización de Técnicas Predictivas y Teoría de Juegos para asignar el resto de bloques de *slots* que quedan sin asignar. Con ello se pretende minimizar el tiempo de transmisión de los mensajes para cada terminal participante en la red, desde que se demandan los *slots* hasta que se transmiten, ya que,

al asignarle más bloques de *slots*, puede transmitir antes en el siguiente *reallocation period*.

Para ello, se han planteado dos propuestas: una propuesta basada en Técnicas Predictivas, que es un método basado en la estimación de la media de la demanda de bloques de *slots* para todos los terminales en los últimos *reallocation periods*, y una propuesta basada en Teoría de Juegos, que es el Dilema del Prisionero Iterado para *N* jugadores, y se ha realizado un estudio comparativo entre ellas y el algoritmo de asignación actual en referencia al tiempo de transmisión de los mensajes del terminal.

Los resultados de las simulaciones muestran que todos los métodos propuestos mejoran el algoritmo TSR actual. El método de estimación de la media es el que mejores resultados obtiene en todos los casos. Además, todos los métodos son bastante rápidos a la hora de converger a la solución, como el algoritmo TSR actual. Estos resultados son independientes del tipo de distribución del tráfico de datos de entrada.

Con todo ello, las principales aportaciones de esta tesis son las siguientes:

- Se propone la mejora del algoritmo TSR en una red Link-16 en el caso de que la demanda total de bloques de *slots* para todos los terminales sea mayor que el número de bloques de *slots* disponible. Estas mejoras consisten en eliminar los casos de conflicto en la asignación y minimizar el valor medio de la cola máxima de mensajes no transmitidos en cada terminal participante para un número de *reallocation periods* dado.

- Para ello, se propone una nueva estrategia: a cada terminal sordo (no recibe ningún mensaje de demanda de *slots* del resto de terminales participantes en la red) se le asigna el número de bloques de *slots* que solicita, y el problema consiste en la asignación del resto de bloques de *slots* al resto de terminales participantes, eliminando el orden de prioridad del algoritmo TSR actual.

- Se propone la utilización de diversas técnicas de Optimización Combinatoria como nuevos algoritmos de reasignación dinámica de *slots* TSR, que mejoran el algoritmo actual. Las propuestas planteadas son: una propuesta basada en el método de

optimización de programación entera Branch and Bound, una propuesta basada en Redes de Hopfield, un algoritmo heurístico y un Algoritmo Genético.

- También se propone la mejora del algoritmo TSR en una red Link-16 en el caso de que la demanda total de bloques de *slots* para todos los terminales sea menor que el número de bloques de *slots* disponible. Estas mejoras consisten en minimizar el tiempo de transmisión de los mensajes para cada terminal participante en la red, desde que se demandan los *slots* hasta que se transmiten, asignando el resto de bloques de *slots* que quedan sin asignar.

- Para ello, se plantean dos propuestas como nuevos algoritmos de reasignación dinámica de *slots* TSR, que mejoran el algoritmo actual. Las propuestas planteadas son: una propuesta basada en la estimación de la media de la demanda de bloques de *slots* para todos los terminales en los últimos *reallocation periods*, y una propuesta basada en Teoría de Juegos, que es el Dilema del Prisionero Iterado para *N* jugadores.

## 9.2 Líneas futuras de trabajo.

Como continuación al trabajo desarrollado en la presente tesis, se proponen las siguientes líneas futuras de trabajo:

- Solicitud de una revisión del estándar Link-16 para dar cabida a los nuevos algoritmos presentados en esta tesis como mejora del algoritmo TSR (*Time Slot Reallocation*) de reasignación dinámica de *time slots*.

- Se propone realizar un estudio de los patrones de tráfico de las peticiones de bloques de *slots* de cada participante en la red Link-16, por si pudiesen asemejarse a una distribución de trafico estandarizada.

- Se propone el estudio de la aplicación de estos algoritmos en el nuevo sistema de enlace de datos tácticos Link-22. Es un sistema eminentemente naval y basado

también en Acceso Múltiple por División en el Tiempo (TDMA). Opera en las bandas de HF (3-30 MHz) y UHF (225-400 MHz), y es interoperable con Link-16 a efecto de catálogo de mensajes, ya que además de sus mensajes propios (serie F), soporta también los mensajes de la serie J de Link-16 [Stanag22].

El sistema Link-22 presenta múltiples capacidades de reasignación de *time slots*. El acceso TDMA dinámico permite que la estructura de la red se pueda modificar sin interrumpir la operación de la red. Esta forma de acceso consiste en que una unidad que requiera una capacidad de recursos de red adicional pueda solicitárselos a las unidades que tienen un exceso de recursos asignados. Es decir, que al igual que en modo TSR de Link-16, el protocolo permite a una unidad determinar sus necesidades adicionales y solicitarlas de forma que cualquier otra unidad que disponga de recursos suficientes pueda transferírselos. Este proceso puede ocurrir sin la intervención de la unidad de gestión de red.

- Finalmente, se propone el estudio de la influencia del uso de estos nuevos algoritmos en una red en la que existan varios sistemas de enlace de datos tácticos, además de Link-16, más antiguos que éste, como puede ser el Link-4 o el Link-11.

  Link-4 es un sistema de enlace de datos que proporciona vectores de mando a las aeronaves que requieran de control externo. Es un enlace no seguro en la banda UHF y la tasa de datos es de 5000 bps [Stanag04].

  Link-11 es un sistema de enlace de datos eminentemente naval. Está basado en tecnología de los años 60 y tiene una tasa binaria relativamente pequeña: 1364 bps a 2250 bps. El acceso se realiza por sondeo a cada unidad participante (modo *Roll-Call*) y es necesario disponer de una estación de control de la red (NCS, *Net Control Station*) [Stanag11]. Está previsto que el sistema Link-22 reemplace a Link-11 en un plazo de 5 años aproximadamente.

# Parte V

# Apéndice

# Apéndice A Algoritmo de asignación dinámica de slots TSR

## A.1 Introducción.

En este apéndice se describe el algoritmo de selección y asignación dinámica de *time slots* para el modo de acceso TSR (*Time Slot Reallocation*) en una red de comunicaciones Link-16. En este modo de acceso se asigna un mismo conjunto de *time slots* a varios terminales participantes en la red para la transmisión de sus mensajes [Barto91], pero se van distribuyendo dinámicamente de forma automática a cada unidad de acuerdo a sus requisitos de transmisión en cada momento, es decir, cada participante recibe un número de *slots* de ese conjunto según sean sus necesidades en cada momento.

Este algoritmo de asignación de *slots* reside en cada terminal de la red con capacidad TSR. Cuando utiliza esta capacidad, cada terminal determinará sus necesidades de *time slots* para cada *reallocation periods*, y enviará un mensaje de Link-16, llamado J0.7, con el porcentaje del conjunto de *time slots* disponible para TSR que se desea usar en el siguiente *reallocation period*. Cuando la demanda total de *slots* para todos los terminales sea mayor que el conjunto de *slots* disponible, el algoritmo seleccionará y asignará a cada terminal un número de *slots* proporcional al número demandado.

## A.2 Conceptos básicos.

En el modo de acceso TSR el tiempo se divide en periodos, llamados *reallocation periods* [MIDSCO99]. El tamaño de cada *reallocation period* se fija mediante un parámetro adaptable al realizar el diseño de la red Link-16, y su valor puede variar entre 6 y 48 segundos. Su valor por defecto es 12 segundos, que es la unidad de diseño de redes Link-16.

La unidad de asignación de *time slots* en el modo de acceso TSR es el bloque básico de *slots*, de forma que a cada terminal se le asigna en cada *reallocation period* un número de bloques básicos de *slots* para realizar sus transmisiones en el conjunto de *slots* disponible para este modo de acceso. Este conjunto de *slots* disponible tendrá un máximo de 512 bloques básicos. El número de *slots* que componen un bloque básico se obtiene a partir de una serie de parámetros adaptables [MIDSCO99], cuyos valores se fijan al realizar el diseño de la red Link-16.

En cada *reallocation period*, cada terminal transmite un mensaje J0.7 con su demanda de bloques básicos de *slots* para el siguiente *reallocation periods*. Además, envía en este mensaje la demanda de bloques básicos de *slots* del resto de terminales de la red de los cuales haya recibido su mensaje J0.7 hasta ese momento, de forma que se va diseminando sobre la red la demanda de todos los terminales. Cada terminal crea y mantiene una tabla con la demanda de bloques básicos de *slots* del resto de terminales de la red. Esta tabla se actualiza a medida que se reciben mensajes J0.7. El número de mensajes recibidos por cada terminal del resto de participantes hasta el momento de la transmisión de su mensaje en cada *reallocation period*, se conoce como *receive count* del terminal, parámetro que usará el algoritmo de asignación de *slots*. Los terminales cuyo *receive count* es cero se conocen como terminales sordos, ya que no reciben ningún mensaje de demanda de *slots* de ningún participante.

Todos los mensajes J0.7 enviados en un *reallocation period* tienen que ser recibidos por los terminales antes del llamado *freeze point*, que es 3 segundos antes del final del *reallocation period*. Después de este momento, el algoritmo de asignación de *slots* selecciona y asigna los bloques básicos de *slots* que corresponden a cada terminal

en el siguiente *reallocatiom period* a partir de la información obtenida de los mensajes J0.7 recibidos. Este algoritmo es igual para todos los terminales, y reside en cada terminal.

## A.3 Cálculo de demanda de slots.

En cada *reallocation period*, cada terminal transmite un mensaje J0.7 que incluye su demanda de bloques básicos de *slots* para el siguiente *reallocation period*. El número de bloques demandado dependerá del número de mensajes que el terminal necesite transmitir en el siguiente *reallocation period*. La demanda de *slots* se indica como porcentaje del conjunto de *time slots* disponible para TSR.

Para determinar su demanda de *slots* para el siguiente *reallocation period*, cada terminal necesita los siguientes parámetros, que se fijan al realizar el diseño de la red Link-16:

- *H*: Número de mensajes que se predice transmitir durante el siguiente *reallocation period*.
- *W*: Número medio de palabras por mensaje a transmitir.
- *L*: Número máximo de palabras que se pueden transmitir en un *time slot*.
- *B*: Número de *time slots* que componen un bloque básico.
- *D*: Número de *time slots* necesarios para poder transmitir el mensaje J0.7 en el siguiente *reallocation period*.
- *P*: Número de bloques básicos en el *reallocation period*.

Con estos parámetros, el terminal calcula el número de bloques básicos de *slots* que necesita, *T*, usando la ecuación:

$$T = \frac{\frac{H \times W}{L} + 1}{B} \qquad (A.1)$$

Si el terminal no recibe valores para *H* y *W*, pone *T* = 1.

El porcentaje de *slots* demandado por el terminal, $F$, tiene un conjunto de valores posibles restringido a los valores permitidos en el campo correspondiente de la palabra correspondiente del mensaje J0.7 [MIDSCO99]. El máximo porcentaje que un terminal puede demandar es el 60 %. Con todo ello, el terminal comprueba el valor del parámetro:

$$S = \frac{T - \frac{D}{B}}{P} \qquad (A.2)$$

que es el porcentaje de bloques básicos de *time slots* que el terminal necesita, descontando el número de bloques básicos de *time slots* necesarios para poder transmitir el mensaje J0.7 en el siguiente *reallocation period*.

Si $S \leq 0.6$, el terminal determina el valor más pequeño de los posibles para $F$ que cumpla que:

$$nin(F \times P) \geq T - \frac{D}{B} \qquad (A.3)$$

donde $nin(X)$ es el entero más próximo a $X$.

Si $S > 0.6$ y el valor del *receive count* del terminal es distinto de 0 (terminal no sordo), toma $F = 0.6$.

Si $S > 0.22$, el valor del *receive count* del terminal es 0 (terminal sordo) y el parámetro *Request Limit Override* está deshabilitado (este parámetro indica si el terminal puede pedir un porcentaje mayor del calculado), toma $F = 0.22$.

Finalmente, si $S > 0.6$, el valor del *receive count* del terminal es 0 (terminal sordo) y el parámetro *Request Limit Override* está habilitado, toma $F = 0.6$.

## A.4 Asignación y selección de bloques básicos de slots.

Después del *freeze point* en cada *reallocation period*, cada terminal selecciona los bloques básicos de *slots* que se asigna para el siguiente *reallocation period*, calculados mediante el algoritmo de asignación. Para ello utiliza sus datos y los del resto de participantes en la red, que mantiene en la tabla actualizada con los datos del resto de participantes. En esta tabla, cada participante, con sus datos, se referencia mediante su posición en la tabla o bien mediante su parámetro *Source Track Number*, que es un número en octal que se asocia a cada participante en una red Link-16.

En el algoritmo de asignación, cada terminal asigna una prioridad a cada participante conocido usando el *receive count* del participante, guardado en la tabla con los datos de los participantes. La prioridad aumenta a medida que decrece el *receive count*. Para participantes con el mismo *receive count*, la prioridad es mayor para los participantes referenciados mediante su posición en la tabla frente a los referenciados mediante su *Source Track Number*. Y para participantes con el mismo modo de referenciarse, la prioridad es mayor para los participantes con menor número (de posición o de *Source Track Number*).

Los terminales cuyo *receive count* es cero se llaman terminales sordos. No reciben ningún mensaje de demanda de *slots*, por lo que no tienen conocimiento alguno de la demanda del resto de terminales. Por eso, estos terminales se asignan todos los bloques básicos de *slots* que solicitan.

Para el resto de terminales, cada uno de ellos estima la selección de bloques básicos de *slots* de todos los terminales sordos conocidos. El terminal emula cada uno de estos terminales sordos asignando todos los bloques básicos de *slots* que solicitan, permitiendo conflictos de selección de bloques entre los terminales sordos. Después de esto, calcula los siguientes parámetros:

- *Q*: Número total de bloques básicos de *slots* menos la suma de los bloques básicos asignados a todos los terminales sordos.

- $M_1$: Suma de las demandas de bloques básicos de *slots* de todos los terminales no sordos cuyo *receive count* es menor que el del propio terminal.
- $M_2$: Suma de las demandas de bloques básicos de *slots* de todos los terminales no sordos cuyo *receive count* es menor o igual que el del propio terminal.

Si $M_2 \leq Q$, el terminal se asigna a él y a todos los terminales cuyo *receive count* es menor o igual que el del propio terminal, todos los bloques básicos de *slots* solicitados.

Si $M_1 < Q < M_2$, el terminal calcula $Q_1 = Q - M_1$ y $M_R = M_2 - M_1$. Se observa que $Q_1$ es el número de bloques básicos de *slots* disponible para todos los terminales con igual *receive count* que el propio terminal, y $M_R$ es la suma de las demandas de todos los terminales con igual *receive count* que el propio terminal.

Sea $R_O$ el número de bloques básicos de *slots* demandados por el terminal, y $N$ el número de terminales con igual *receive count* que el propio terminal, incluyéndole a él. Entonces, si $Q_1 \geq N$, el terminal se asigna un número de bloques básicos de *slots* que viene dado por la ecuación:

$$R_S = integer\frac{(R_O - 1)(Q_1 - N)}{M_R - N} + 1 \qquad (A.4)$$

donde *integer*($X$) es la parte entera de $X$. Si $Q_1 < N$, los $Q_1$ terminales de mayor prioridad seleccionan un bloque básico de *slots* de forma pseudoaleatoria. Los restantes $N - Q_1$ terminales seleccionan un bloque básico de *slots* de forma secuencial usando los parámetros del terminal $Q_1$ +1.

Finalmente, si $Q \leq M_1$, cada terminal con igual *receive count* que el propio terminal selecciona un bloque básico de *slots* secuencialmente usando los parámetros del terminal de mayor prioridad.

Una vez asignado el número de bloques básicos de *slots*, cada terminal puede seleccionar sus bloques o los de cualquier otro participante en el conjunto de *slots* disponible para TSR. Para ello, el terminal usa la posición en su tabla o el *Source Track*

*Number* de ese participante, y el número de *reallocation period*. Ya que el proceso completo de selección de bloques no es relevante para el contenido de esta tesis, que se basa en la asignación de un número de bloques básicos de *slots* a cada terminal participante, se remite al lector al documento [MIDSCO99], donde se explica el proceso de forma detallada.

# Bibliografía


[Aarts89] E. Aarts, J. Korst, "Simulated Annealing and Boltzman Machines", New York: Wiley, 1989.

[Abe89] S. Abe, "Theories on the Hopfield Neural Networks", *Proc. of the Int. Joint Conf. on Neural Networks*, vol. 1, pp. 557-564, June, 1989.

[ADatP97] "Allied Data Processing Publication for Link-16 (ADatP-16)", April, 1997.

[Aiyer90] S. V. B. Aiyer, M. Niranjan and F. Fallside, "A Theoretical Investigation into the Performance of the Hopfield Model", *IEEE Transactions on Neural Networks*, vol. 1, nº 2, June, 1990.

[Akay94] M. Akay, *Biomedical Signal Processing*, Academic Press, San Diego (CA), USA, 1994.

[Ali93] M. K. Ali, F. Kamoun, "Neural Networks for Shortest Path Computations and Routing in Computer Networks", *IEEE Trans. on Neural Networks*, vol. 4, nº 6, pp. 941-953, November, 1993.

[Alspector93] J. Alspector, R. Goodman, T. X. Brown (Eds.), *Proc. of the Int. Workshop on Applic. of Neural Networks to Telecommunications*, Hillsdale, NJ: Lawrence Erlbaum, 1993.

[Amin94] S. Amin and M. Gell, "Investigation of the Hopfield Model and its Attractors", *Neural Comput. and Applic.*, vol. 2, pp. 129-133, 1994.

[Axelrod84] R. Axelrod, *The Evolution of Cooperation*, Basic Books, New York (NY), USA, 1984.

[Bankes94] S. Bankes, "Exploring the Foundations of Artificial Societies: Experiments in Evolving Solutions to Iterated N-Player Prisoner´s Dilemma", *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, MA. MIT Press, 1994.

[Barto91] Jon L. Barto, "Air Force JTIDS Network Library Design Guide", September, 1991.

[Bousoño95] C. Bousoño-Calzón, M. Manning, "The Hopfield Neural Networks Applied to the Quadratic Assignment Problem", *Neural Computing and Applications*, vol. 3, pp. 64-72, March, 1995.

[Bousoño1] C. Bousoño-Calzón, A. R. Figueiras-Vidal, "Competitive Routing of ATM Permanent Virtual Circuits with a Hopfield Based Neural Network", submitted to *IEEE Journal on Selected Areas in Comm.*

[Burshtein94] D. Burshtein, "Nondirect Convergence Radius and Number of Iterations of the Hopfield Associative Memory", *IEEE Trans. on Inf. Th.*, vol. 40, nº 3, pp. 838-847, May, 1994.

[Chakraborty01] G. Chakraborty, "An Efficient Heuristic Algorithm for Channel Assignment Problem in Cellular Radio Networks", *IEEE Trans.Veh. Technol.*, vol. 50, pp. 1528-1539, November, 2001.

[Cichocki93] A. Cichocki, R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, New York: Wiley, 1993.

[Coolen93] T. Coolen, D. Sherrington, "Dynamics of Attractor Neural Networks", en J. G. Taylor (Ed.), *Mathematical Approaches to Neural Networks*, The Netherlands: Elsevier, 1993.

[Davidor91] Y. Davidor, "Epistasis Variance: A Viewpoint on GA-hardness". In J. E. Rowling, editor, *Foundations of Genetic Algorithms I*, San Mateo, CA: Morgan Kaufmann, 1991.

[Fletcher87] R. Fletcher, *Practical Methods of Optimization*, Wiley, 1987.

[Fritsh92] T. Fritsh, M. Mittler, P. Tran-Gia, "Artificial Neural Net Applications in Telecommunication Systems", Research Report, Institute of Computer Science, University of Wuerzburg, 1992.

[Funabiki92a] N. Funabiki, Y. Takefuji, "A Parallel Algorithm for Channel Routing Problems", *IEEE Trans. on Computer-Aided Design*, vol. 11, pp. 464-474, April, 1992.

[Funabiki92b] N. Funabiki, Y. Takefuji, "A Neural Network Parallel Algorithm for Channel Assignment Problems in Cellular Radio Networks", *IEEE Trans. on Veh. Technol.*, vol. 41, nº 4, pp. 431-437, November, 1992.

[Funabiki93] N. Funabiki, Y. Takefuji, "A Neural Network Approach to Topological Via Minimization Problems", *IEEE Trans. on Computer-Aided Design*, vol. 12, pp. 770-779, June, 1993.

[Funabiki95] N. Funabiki, S. Nishikawa, "An Improved Neural Network for Channel Assignment Problems in Cellular Mobile Communication Systems", *IEICE Trans. on Commun.*, vol. E78-B, nº 8, pp. 1187-1196, August, 1995.

[Funabiki97] N. Funabiki, S. Nishikawa, "A Binary Hopfield Neural Network Approach for Satellite Broadcast Scheduling Problems", *IEEE*

*Transactions on Neural Networks*, vol. 8, nº 2, pp. 441-445, March, 1997.

[Goldberg89] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wesley, 1989.

[Grotschel93] M. Grotschel, L. Lovasz, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Berlin: Springer-Verlag, 1993.

[Haykin99] S. Haykin, *Neural Networks. A Comprehensive Foundation, 2ª Edition*, Prentice Hall International, 1999.

[Heckendorn99] R. B. Heckendorn, D. Whitley, "Predicting Epistasis from Mathematical Models", *Evolutionary Computation*, vol. 7, nº 1, pp. 69-101, 1999.

[Hertz91] J. Hertz, A. Krogh, R. G. Palmer, *Introduction to the Theory of Neural Computation*, Redwood City, CA: Addison-Wesley, 1991.

[Hinterding95] R. Hinterding, H. Gielewski, T. C. Peachey, "The Nature of Mutation in Genetic Algorithms", *Proc. 6th International Conference on Genetic Algorithms*, pp. 65-72, 1995.

[Holland75] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, USA, 1975.

[Hopfield82] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Natl. Acad. Sci. USA, Biophysics*, vol. 79, pp. 2554-2558, April, 1982.

[Hopfield85] J. J. Hopfield, D. W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biol. Cybern.*, nº 52, pp. 141-152, 1985.

[IEEE00] *IEEE J. Select. Areas Commun.*, vol. 18, no. 12, December, 2000.

[Jong92] K. A. de Jong, W. M. Spears, "A Formal Analysis of the Role of Multi-point Crossover in Genetic Algorithms", *Annals of Mathematics and Artificial Intelligence*, vol. 5, nº 1, pp. 1-26, 1992.

[Kohonen88] T. Kohonen, *Self-Organization and Associative Memory*, Berlin: Springer-Verlag, 1988.

[Koza92] J. R. Koza, *Genetic Programming*, The MIT Press, Cambridge (MA), USA, 1992.

[Kurokawa94] T. Kurokawa, H. Yamashita, "Bus connected neural network hardware system", *Electron. Lett.*,vol. 30, nº 12, pp. 979-980, June 1994.

[Lai96] W. K. Lai, G. G. Coghill, "Channel assignment through evolutionary optimization", *IEEE Trans.Veh. Technol.*, vol. 45, nº 1, pp. 91-95, 1996.

[Lazaro00] O. Lazaro, D. Girma, "A Hopfield neural network based dinamic channel allocation with handoff channel reservation control", *IEEE Trans.Veh. Technol.*, vol. 49, nº 5, pp. 1578-1587, September, 2000.

[LiJH88] J. H. Li, A. N. Michel, W. Porod, "Qualitative Analysis and Synthesis of a Class of Neural Networks", *IEEE Trans.Circuits Syst.*, vol. 35, nº 8, pp. 976-985, August, 1988.

[LiT90] T. Li, "A Comparative Study of Competition Based and Mean Field Networks Using Quadratic Assignment", Proc. of 2nd INSTED Int. Symposium, *Expert Systems and NN*, pp. 23-26, Haway, 1990.

[Logicon94] Logicon, Inc. Tactical & Training Systems, "Understanding Link-16. A Guidebook for New Users", April, 1994.

[Mangat95] John Mangat-Rai, "JTIDS Network Design and System Management Course Documentation", August, 1995.

[Mangat97] John Mangat-Rai, Skip Fumia, "Link-16 Network Design and System Management Course", 1997.

[McCulloch43] W. S. McCulloch, W. H. Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity", *Bull. Math. Biophys.*, vol. 5, p. 115, 1943.

[McEliece87] R. J. McEliece, E. C. Posner, E. R. Rodemich, S. S. Veukatosh, "The Capacity of the Hopfield Associative Memory", *IEEE Trans. on Inf. Th.*, vol. 33, nº 4, pp. 461-482, July, 1987.

[Michalewicz92] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, New York (NY), USA, 1992.

[Michel89] A. N. Michel, J. A. Farrell, W. Porod, "Qualitative Analysis of Neural Networks", *IEEE Trans.Circuits Syst.*, vol. 36, nº 2, pp. 229-243, February, 1989.

[MIDSCO99] MIDSCO, Inc, "System Segment Specification for the MIDS Low-Volume Terminal and Ancillary Equipment", April, 1999.

[Minoux86] M. Minoux, *Mathematical Programming*, Chichester, UK: Wiley, 1986.

[Movahhedinia95] N. Movahhedinia, G. Stamatelos, H. M. Hafez, "A Slot Assignment Protocol for Indoor Wireless ATM Networks Using the Channel Characteristics and the Traffic Parameters", *IEEE Proc. Globecom 95*, pp. 327-331, November, 1995.

[Nash50] J. F. Nash, "Equilibrium Points in N-Person Games", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 36, pp. 48-49, 1950.

[Nash51] J. F. Nash, "Non-Cooperative Games", *Annals of Mathematics*, vol. 54, pp. 286-295, 1951.

[Neumann44] J. von Neumann, O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, Princeton, 1944.

[Papadimitriou82] C. H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization*, Englewood Cliffs, NJ: Prentice-Hall, 1982.

[Peterson89] C. Peterson, B. Soderberg, "A New Method for Mapping Optimization Problems onto Neural Networks", *International Journal of Neural Systems*, vol. 1, nº 1, pp. 3-22, 1989.

[Rhee91] W. T. Rhee, "Stochastic Analysis of the Quadratic Assignment Problem", *Mathematics of Operations Research*, vol. 16, nº 2, pp. 16-25, May, 1991.

[Rong91] L. Rong, L. Ze-min, "Determination of the Parameters in a Hopfield-Tank Model on Solving TSP", *IEEE Int. Joint Conf. on Neural Networks*, vol. 3, pp. 2455-2460, November, 1991.

[Schelling78] T. Schelling, *Micromotives and Macrobehavior*, New York: W. W. Norton and Co., 1978.

[Sherrington93] D. Sherrington, "The Spin Glass Approach", en J. G. Taylor (Ed.), *Mathematical Approaches to Neural Networks*, The Netherlands: Elsevier, 1993.

[Stanag04] "NATO Standarization Agreement (STANAG) 5504".

[Stanag11] "NATO Standarization Agreement (STANAG) 5511".

[Stanag16] "NATO Standarization Agreement (STANAG) 5516".

[Stanag22] "NATO Standarization Agreement (STANAG) 5522".

[Uesaka91] Y. Uesaka, "Mathematical Aspects of Neuro-Dynamics for Combinatorial Optimization", en T. Kohonen, K. Mkisara, O. Simula and J. Kangas (Eds.), *Artificial Neural Networks*, North-Holland: Elsevier, 1991.

[Vera02] A. Vera, A. Artés, "Improved Time Slot Reallocation (TSR) Algorithms in Link-16 Communications Networks", *Learning 02*, Universidad Carlos III, Madrid, 2002.

[Vera05] A. Vera, "Manual de Usuario de la aplicación A.N.I.T.A. versión 4.0", Enero, 2005.

[Zweben94] M. Zweben, M. Fox, *Intelligent Scheduling*, Morgan Kaufmann Publishers, San Francisco (CA), USA, 1994.